University of Warsaw
Faculty of Economic Sciences

Working Papers
No. 18/2020 (324)

# TOWARDS BETTER UNDERSTANDING OF COMPLEX MACHINE LEARNING MODELS USING EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI) - CASE OF CREDIT SCORING MODELLING

Marta Kłosok
Marcin Chlebus

Warsaw 2020

WORKING PAPERS 18/2020 (324)

UNIVERSITY OF WARSAW
FACULTY OF ECONOMIC SCIENCES
WORKING PAPERS

# Towards better understanding of complex machine learning models using Explainable Artificial Intelligence (XAI) - case of Credit Scoring modelling

**Marta Kłosok, Marcin Chlebus***

*Faculty of Economic Sciences, University of Warsaw*
** Corresponding author: mchlebus@wne.uw.edu.pl*

**Abstract:** recent years many scientific journals have widely explored the topic of machine learning interpretability. It is important as application of Artificial Intelligence is growing rapidly and its excellent performance is of huge potential for many. There is also need for overcoming the barriers faced by analysts implementing intelligent systems. The biggest one relates to the problem of explaining why the model made a certain prediction. This work brings the topic of methods for understanding a black-box from both the global and local perspective. Numerous agnostic methods aimed at interpreting black-box model behavior and predictions generated by these complex structures are analyzed. Among them are: Permutation Feature Importance, Partial Dependence Plot, Individual Conditional Expectation Curve, Accumulated Local Effects, techniques approximating predictions of the black-box for single observations with surrogate models (interpretable white-boxes) and Shapley values framework. Our prospect leads toward the question to what extent presented tools enhance model transparency. All of the frameworks are examined in practice with a credit default data use case. The overview presented prove that each of the method has some limitations, but overall almost all summarized techniques produce reliable explanations and contribute to higher transparency accountability of decision systems.

**Keywords**: machine learning, explainable Artificial Intelligence, visualization techniques, model interpretation, variable importance

**JEL codes**: C25

## 1. Introduction

Machine learning (ML) has been a powerful tool designed for many useful purposes. Many different books and articles present different definitions of it. Shai Shalev-Shwartz and Shai Ben-David (2014) state that machine learning refers to the automated detection of meaningful patterns in data. Weng (2019) defines it as an approach towards building intelligent machines for Artificial Intelligence (AI). More precisely, machine learning is a branch of Artificial Intelligence (AI), presumably the most popular and widespread field of it, however Artificial Intelligence comprises of much more fields of study than just machine learning. It is pattern recognition, logic-based AI, search, knowledge representation, genetic programming, heuristic and others (Joshi, 2017).

With increasing amount of data available popularity of machine learning algorithms has also increased. One could also say that nowadays it is hardly possible to analyze huge datasets only with the help of a single man. Even if it is, it would take a long time and could turn out to be useless. And now comes machine learning. It enables to analyze and process huge information about human behavior, human tastes, preferences. Mobile phones' are designed to detecting human faces or fingerprints and unlocking themselves. Banks exploit machine learning for the purpose of defining creditworthiness of clients or detecting frauds. These are only few examples of using machine learning in the real world.

Machine learning is about "making machines think like humans" (Joshi, 2017). Algorithms built through this concept are known to bring us with highly accurate predictions. Prediction is just a single result of the system based on a given set of information (features). However machine learning algorithms are not always as rational and intuitive as human beings. They are just machines built in some definite way, often with strong mathematics standing behind. In addition to that, they exploit wide range of statistic and programming concepts. Key idea that separates simple statistical linear models (such as logistic regression) from the complex ML models is their nested non-linear structure (Samek et al., 2017). It is customary to term these structures as black-box algorithms - black-box because no one is fully able to explain how these structures accomplish their results and what features stand behind the predictions. In other words, it is very challenging to understand inner-working of the machine learning systems (Honegger et al., 2018). Here comes another important distinction between machine learning and statistical models. The latter method allows to fit a project-specific probability model of a defined form. It usually requires fulfilling specified assumptions such as normal distribution or equal variance of variables. Finally it is possible to explain how distinct features influence final predictions. On the other hand, ML methods find pattern in rich and unwieldy data with minimal assumptions behind (Bzdok et al., 2018). As a result, they end up with very complex form very difficult to peek inside. Recent years have brought us with a wide range of method trying to investigate Artificial Intelligence predictions.

Interpretability of machine learning algorithms is especially important for decision makers who rely upon analytics and data scientists building sophisticated systems. Systems built on the machine learning foundations often earn huge money for companies. In case the systems break or earn no more money decision makers might want to explain why things go wrong and what can they do in order to improve the systems. Another group of people who might consider explainable AI (XAI) methods desired today are bank and financial regulators. Consider a client who was granted a big loan and then defaulted bringing losses for the bank. If the decision was made by system built upon ML foundations, a bank regulator might require explaining what features answered for a high score assigned to the clients. XAI methods are designed not only for explaining decisions and predictions made by the algorithms, but also improving our systems, debugging and fixing mistakes in the data or implementation of the models.

Multiple methods have been already proposed. For example Friedman (2001) introduces Partial Dependence Plot (PDP) as a model agnostic tool for visualizing average predictions of a model along a specified independent feature. It shows whether a relation between a given feature and the outcome is linear, non-linear, monotonic etc. Another useful tool, in some manner similar to PDP is Individual Conditional Expectation (ICE) plot. It does the same as PDP but for individual observation – presents graphical relationship between the outcome and the feature, but for a single instance (Pitkin, 2014). Permutation Feature Importance – measures increase in the prediction error measure after permuting a given variable and calculating predictions with the new dataset. The higher the error increases, the more important the feature is. In other way, this method ranks all of the features by decreasing importance (Gregorutti et al., 2016). Ribeiro et.al (2016) propose LIME - yet another useful technique - deriving from the concept of local surrogate models. Authors emphasize that trusting an individual prediction is as important as trusting a model as a whole. Thus, they contributed an algorithm approximating a single instance with an interpretable model (for example linear regression). Finally Lundberg and Lee (2017) present a unified framework grounded in the game theory for explainable Artificial Intelligence, SHAP (SHapley  Additive exPlanations).

Adoption of XAI techniques can improve understanding complex structure of the black-box inner working. In this work, different frameworks and methods enhancing explainability and transparency of AI models predictions are proposed. Explanation techniques are evaluated in practice with a credit default data use case. It is checked, to what extent do the methods enable understanding evaluated model. How deep are we able to peek into the complex model in order to capture dependencies among data traced by the algorithm. As post-hoc explanation tools are discussed only, all provided methods bring insight into information after the model is trained. Model interpretation is limited to agnostic approach, so universal for any black-box (but also interpretable white-box models) algorithms. This paper contributes to answering the question, to what extend XAI enables understanding black-box models. Do interpretable tools inform about relation between input variables and the outcome only in the global scope (regarding predictions in whole dataset) or locally (approximating a single observation). Among the goals of this work is also finding most informative and valuable framework that can be facilitated into fields where it is crucial to defend decisions made on the grounds on AI, such as credit scoring contributed in the paper. Overall, six approaches incorporating different theory-driven frameworks are discussed. Contribution of this work includes: Permutation Feature Importance, Partial Dependence Plot, Individual Conditional Expectation Curve, Accumulated Local Effects, techniques approximating predictions of the black-box for single observations with surrogate models (interpretable white-boxes) and Shapley values framework. The paper is organized as follows: first chapter discusses methodologies and concepts seeking to bring more transparency and confidence into the black-box models predictions. Second chapter focus is on data and model evaluated for the purpose of investigating and understanding methods discussed in chapter one. Finally results obtained with frameworks implemented in Python packages are presented.

## 2. Methods

This chapter brings the topic of methods available for explaining Artificial Intelligence. What does it exactly mean that the black-box structure is interpretable for a human or that we can easily explain why the model made a certain prediction? There is no standard criteria and definition for sure. Arrieta et al. (2019) underlie there is some urgent need for consensus regarding definition of Explainable Artificial Intelligence. Authors mention also common pitfall when confusing interpretability and explainability terms. While interpretability (often referred as transparency) is the ability to express the model performance understandable for a human being, explainability refers to actions of peeking into the learning algorithm and understanding its inner structure. Once the functioning of a model is understood and inner connections between distinct predictors explored, building trust into the model is the consequence. It is crucial for decision makers in achieving success. Trustworthiness and confidence of the model is essential, as its outstanding performance might be only a matter of a few patterns found in the small subset of data. Interpretable techniques explored in this paper focus only on post-hoc explanations. For their evaluation a set of explanatory variables, prediction function (learning model generating predictions) and the target outcome is required (Arya et al., 2019). Afterwards each of the explanation method is applied. XAI frameworks are designed to trigger explanations from the global and local scope. The former concerns treating the model as a whole and explaining its outcomes across entire partitions of data. It not only visits significant features in the model, but also looks for prediction sensitivity with respect to single predictor values and interactions within two features. However these global relations often are measured in some approximate terms, such as average influence of the input variable for the output etc. Local explanations focus on analyzing predictions of single instances or promoting small subsets of the observations located in close neighborhood of each other. Usually purpose of the local method is to decompose prediction of the model for a given instance with  features contributions for each of the component (Hall and Gill., 2018). In this paper main focus is on model agnostic tools allowing inspecting any kind of machine learning model. In addition to that, most of the tools offer ability to explain more complicated deep learning algorithms implemented e.g. for image recognition. It is believed this approach is most useful as model agnostic tools can be applied to wide range of methods implemented through different packages and systems. These agnostic tools are often implemented in different programming languages in a similar approach, as theory standing behind them does not change. Model specific tools, on the other hand, are often developed on the grounds of a single implementation of the model that might differ among different computer systems.

### *2.1. Permutation feature importance*

At first, a single representation of a model agnostic method is described. Variable importance was first introduced by Breiman (2001) in a random forest algorithm. Further research was done by Fisher et al. (2018) who proposed model agnostic tool for calculating contribution of individual features into prediction accuracy based on Breiman (2001) approach. Variable importance is calculated by randomly permuting a variable $X_i$ and computing increase in prediction error with the newly created learning sample.

Denote vector of $k$ independent variables $X = (X_1, X_2, \dots, X_k)$, $Y = (Y_1, Y_2, \dots, Y_k)$ is a vector of true outcomes (dependent variable) and so $D_n = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ is a training set for the learning model. Once prediction function of a model $\hat{f}(x)$ is obtained, prediction error of the algorithm is denoted as $R(\hat{f}) = E[(\hat{f}(X) - Y)^2]$. Instead of compounding term – absolute error, root mean squared error or any other metric such as R-squared, adjusted R-squared, accuracy, Gini measure or any other scoring function incorporating predicted and true outcomes might be used in order to calculate performance of the model.

Suppose an ensemble learning algorithm (e.g. random forest) composed of $n$ trees is trained. Because each of decision trees is built on different sample being only a fraction of the whole training set $D_n$, prediction function $R(\hat{f})$ cannot be computed directly. Hence the following estimator is proposed, where $\bar{D}$ denotes a validation set and n is the number of observations in the validation sample.

$$\hat{R}(\hat{f}, \bar{D}) = \frac{1}{n_{tree}} \sum_{t=1}^{n_{tree}} (\hat{f}_t(X_{\mathrm{i}}) - Y_{\mathrm{i}})^2, i = 1,2, \dots, n \tag{1}$$

Function returning prediction error is aggregation of predictions returned by all of the ensemble tree estimators $\hat{f}_1, \hat{f}_2, \dots, \hat{f}_{n_{tree}}$ into the average value. Alternatively for a model consisting of a single structure(e.g. logistic regression) $n_{tree}$ is set to one and estimator $\hat{R}(\hat{f}, \bar{D})$ can be derived in a similar manner.

Once error measure function and original model error is obtained it is possible to formally define Permutation Feature Importance (PFI) of variable $X_i$. Given an original learning sample $D_n$, permute a feature $X_i$ by shuffling its values randomly. Instead of rearranging the values creating a new variable with a given distribution is also acceptable. After permuting the variable once again calculate prediction error with this newly created dataset denoted as $D_n'$.

$$Imp(X_{\mathrm{i}}) = \frac{1}{n_{tree}} \sum_{t=1}^{n_{tree}} (\hat{R}\left(\hat{f}_t, \overline{D_n^{t'}}\right) - \hat{R}(\hat{f}_t, \overline{D_n^{t}})) \tag{2}$$

Finally variable importance for predictor $X_{\mathrm{i}}$ is the difference between prediction error of the estimator calculated with the shuffled data and prediction error with the original data. In order to compare significance of all training features $Imp(X_{\mathrm{i}})$ should be computed for all of

$i = 1, \ldots, k$ features and sorted by descending values. Above formula state that there are $n_{tree}$ tree-based algorithms creating ensemble model (Gregorutti et al., 2016).

There are some concerns regarding this method. First touches the issue whether importance should be calculated with training or testing sample. For the purpose of this paper importance will be computed with training sample, as it is used for model training and optimal parameter search. According to best practices the test set should serve only for the purpose of checking performance of the algorithm in terms of accuracy of predictions and so it is done here. Secondly, permutation of predictor values is arbitrary in every iteration. As a consequence, shuffling the feature for the first time might bring us with different importance than for the second time. Solution for eliminating this randomness component might be performing the action many times and taking the average result. Plotting standard deviation of calculations is also desirable. In addition to that, variable significance can be measured trustworthily only if the predictors are independent of each other. However it is common in practice that independent variables are strongly correlated. This doubt was raised by Tolosi and Lengauer (2011) who noticed that permutation feature importance depends strongly on the correlation between predictors. They contribute that features belonging to larger group of correlated features receive smaller importance weights even though they might significantly correlate with the dependent variable. Lastly, consider learning algorithm performing very well on the training sample. It can be so good that we can speak in terms of overfitting to the data. Such a machine learning structure might be missing some data properties when tested with the new data sample and PFI calculated on the testing set might completely different than PFI with training set. Mentch  and Hoker (2019) raise arguments against this technique more in detail. They mention also two meaningful alternatives. The first one suggested primarily by Strobl et al. (2008) involves permuting predictor taking into consideration distribution conditional on the other regressors. Second method studied by Lehmann and Romano (2006) is leave-one-covariate-out (LOCO). In line with LOCO Feature Importance should be calculated by measuring predictor error after calculating the model without feature of interest. Nevertheless, all these methods seem to produce comparable results (Mentch  and Hoker, 2019). Yet another alternative of calculating Feature Importance is provided in Shapley values subchapter.

## *2.2 Partial Dependence Plot*

Zhao and Hastie (2018) state that Partial Dependence Plot (PDP) depicting marginal effect of an input variable and the model outcome is a black-box visualization tool most commonly used.

This approach was primary introduced by Friedman (2001). With PDP relation between a model outcome and the feature values might be visualized on the plot and better understood. This graphical representation of predictions is another agnostic tool for global model inspection. What PDP does for a given variable is averaging model predictions keeping $X_i$ feature values constant. This toolbox is especially useful when direction of regressors influence on the model outcome is unknown and its assessment is required.

Writing the procedure formally: let $X = (X_1, X_2, \dots, X_k)$ represent a vector of $k$ variables and there are $n$ observations in the learning sample. Suppose $x_s$ represents an interest set for which PDP is to be calculated and $x_c$ is its compliment. Compliment $x_c$ refers to the observations not included in set $x_s$, so $x_c = x \setminus x_s$. Prediction function of a model on dataset $x$ is denoted as $\hat{f}(x)$. Partial dependence function on a set $x_s$ is defined as:

$$f_{PD}(x_s) = E_{x_c}\left[\hat{f}(x_s, x_c)\right] = \int \hat{f}(x_s, x_c)\, dP(x_c) \tag{3}$$

where $dP(x_c)$ refers to the marginal distribution of $x_c$. Since it is not precisely known, Partial Dependence might be computed with the formula:

$$\overline{f_{PD}}(x_s) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}\left(x_s, x_{i,c}\right) \tag{4}$$

What this method does is simply averaging over features values keeping $x_s$ constant (Greenwell, 2017)

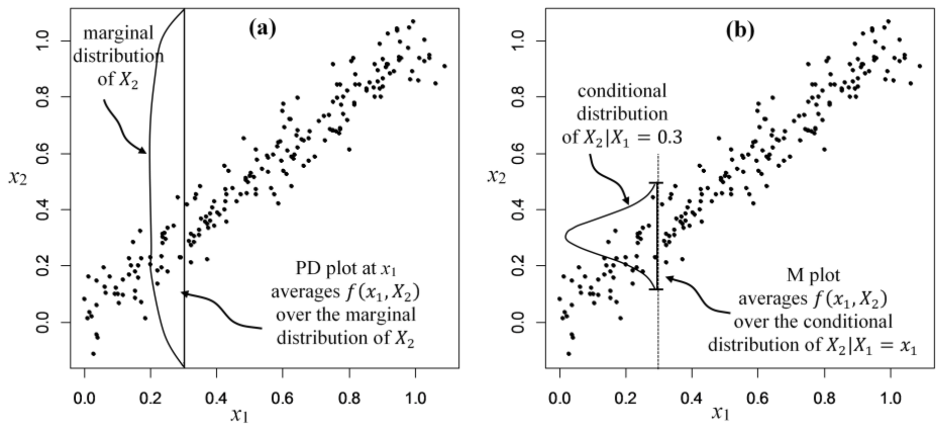
Idea behind constructing PD estimator is rather simple. Suppose we have the training set $D_n = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. First step in the calculations involves replacing all of the data points from set $D_n$ with the value for the first observation of that feature. Afterwards, predictions with newly created dataset (keeping $x_{1i}$ constant) are obtained and average value of the predictions is calculated. The procedure is repeated by replacing all of the feature values with the second observation value (keeping $x_{2i}$ constant) and again average prediction is obtained. Suppose number of unique values for predictor is equal to number of data points $n$. The procedure is repeated then $n$ times until all average effects are obtained. Finally plotting average values calculated for all of $k$-feature values is possible.

Partial Dependence Plots are most popular for covariates in one dimensional space, because plotting them for more dimensions is difficult to analyze, although 2-dimensional PDP visualizing interactions between explanatory variables are popular too. Jerome (2001) suggests that with large number of predictors it is recommended to have a measure of relevance such as permutation feature importance, because selecting a couple of useful variables and their combinations might be time consuming. Partial Dependence is also computationally expensive. Consider a feature $j$, number of data points $n$ and number of grid points $m$. For every predictor it is required to make $n * m$ predictions. Supposing there are hundreds of explanatory variables, it is reasonable to select a couple of most desirable features for visualization. Reducing computational burden is possible with setting confidence interval. Instead of selecting the whole range of data grids, one can define quantiles of the feature distribution for PD calculation.

Certainly partial dependence plots the relation between model outcome and input variable both for categorical and continuous variables. The former is possible to obtain by simply plotting a single feature values against $\overline{f_{PD}}(x_s)$. For the latter PD might be presented as a bar plot, where each bar denotes the unique categorical value and height of the bar corresponds to the PD function value (Jerome, 2011). For many unique values of predictor might be computationally expensive to calculate Partial Dependence estimator.

Similarly as with PFI correctness of this method might be questioned in case of highly correlated features. Fortunately an alternative method to partial dependence was proposed. Accumulated Local Effects (ALE) takes the correlation bias into account and with slight modifications calculates average predicted outcome with respect to the predictor value. Another problem with PD is that averaging in equation (4) is computed over marginal distribution of other features. This inflates the results because observations with hardly probable numbers are created, thus creating fake and unrealistic data and inflating Partial Dependence curve shape. The problem is depicted in Figure 1(a).

**Figure 1. Left panel (a) presents calculation of Partial Dependence at $x_1 = 0.3$ taking into account marginal distribution of predictor $X_2$. PD keeps $x_1 = 0.3$ constant for all of the observations creating fake and unrealistic data points. Marginal plot on the right panel (b) averages predictions only for those instances where $x_1 = 0.3$.**



*Source:* Daniel W. Apley and Jingyu Zhu. "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models." *Northwestern University, USA*, 2019.

Plot illustrates two correlated features $X_1$ and $X_2$. What PDP for feature $X_1 = 0.3$ does is calculating its effect on average prediction over marginal distribution of $X_2$. Its consequence is making combinations of nonexistent pairs such as ($X_1 = 0.3$, $X_2 = 0.8$) or ($X_1 = 0.3$, $X_2 = 0$). Alternatively instead of marginal density conditional distribution might be used. In this manner Marginal Plot was proposed. It is illustrated in Figure 1(b). Function calculating MP values takes the form:

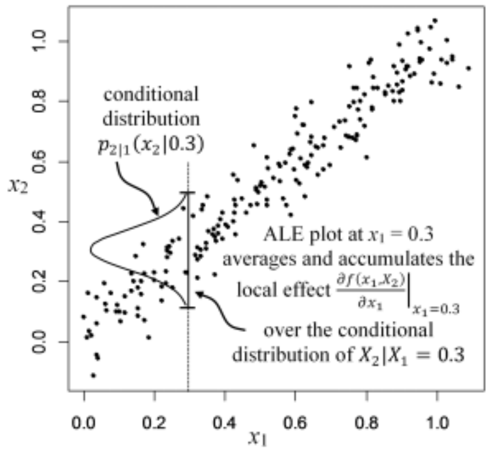
$$f_M(x_s) = E_{x_c|x_s}\left[\hat{f}(X_s, X_c) \middle| X_s = x_s\right] = \int \hat{f}(x_s, x_c)\, dP(x_c|x_s) \tag{5}$$

Analogously estimator of this formula is obtained:

$$\overline{f_M}(x_s) = \frac{1}{n(x_s)} \sum_{i \in N(x_s)} \hat{f}(x_s, x_{i,c}) \tag{6}$$

In order to calculate $\overline{f_M}$ it is necessary to define some neighborhood of the feature of interest $x_s$. $N(x_s) \subset \{1,2,\dots,n\}$ is the subset of observations where $x_{i,s}$ is in close neighborhood of $x_s$ and $n(x_s)$ denotes number of those instances (Apley and Zhu, 2019). Figure 1(b) shows average predicted outcome over conditional density of $X_2$ under condition that $x_1 = 0.3$. Difference between the plots is substantial. Marginal Plot resolves the problem of incorporating marginal distribution, however does not account for correlation bias, as averaging still incorporates dependence between two features. Solution for this phenomena was proposed by Zhu and Apley (2019) and is named as ALE plot.

**Figure 2. Calculation of Accumulated Local Effect at point $x_1 = 0.3$. Conditional distribution is taken into account similarly as in Marginal Plots, however changes of predictions with respect to small changes of predictor $X_2$ are computed in close vicinity of predictor $x_1 = 0.3$.**



*Source:* Daniel W. Apley and Jingyu Zhu. "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models.*" Northwestern University, USA*, 2019.

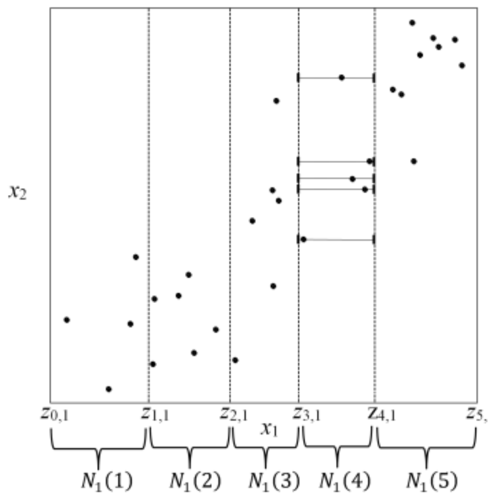
For a given predictor Accumulated Local Effect calculates average changes in prediction for observations in close neighborhood to the original one. Formally ALE requires computing:

$$f_{ALE}(x_s) = \int_{x_{min,x_S}}^{x_S} E_{x_c|x_s}\left[\widehat{f^1}(X_s, X_c)\middle|X_s = z_s\right]dz_s - constant =$$

$$\int_{x_{min,x_S}}^{x_S} \int \widehat{f^1}(z_s, x_c)\mathbb{P}(x_c|z_s)\, dx_c\, dz_s - constant \qquad (7)$$

where $\widehat{f^1}(x_s, x_c) = \frac{\partial \hat{f}(x_s, x_c)}{\partial \hat{f}(x_s)}$ denotes the first derivative - change of $\hat{f}(x_s, x_c)$ with respect to small change of $x_s$ . When calulating local effect of $x_s$ ALE accumulates effect of the prediction for a given predictor with the gradient. For better understanding ALE plot idea Figure 3 with two features $X_1$ and $X_2$ is presented.

**Figure 3. Calculation of Accumulated Local Effect for predictor $X_1$ correlated with $X_2$. There are 30 observations divided into $K = 5$ intervals. It is supposed that intervals ranges denoted as $z_{k,1}$, $k = \{1, 2, 3, 4, 5\}$ are located in close vicinity of each other. Calculation of ALE main effect of the model prediction function $\hat{f}$ within interval $N_1(4)$ consisting of $n_1(4) = 5$ points involves computing prediction for these points pretending their $x_1$ is set to $z_{4,1}$ minus predictions with $x_1 = z_{3,1}$.Differences of predictions for five datapoints are summed and averaged.**



*Source:* Daniel W. Apley and Jingyu Zhu. "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models.*" Northwestern University, USA*, 2019.

Example requires computing ALE with respect to predictor $X_1$. For this purpose dividing the feature of interest in many intervals is needed. In Figure 3 predictor is divided into five partitions only, however in practice sufficiently large number should be taken, depending on the number of observations and unique values of explanatory variable. Intervals are designated as $N_j(k)$ and number of instances falling into $k - th$ interval is denoted as $n_j(k)$, where $j = \{1,2,..,\mathrm{d}\}$ denotes the feature index and $k = \{1,2,..,\mathrm{K}\}$ denotes the interval number. Ranges of intervals are set with following notation: $N_j(k) = (z_{k-1,j}, z_{k,j}]$. Authors Zhu and Apley state that $z_{k,j}$ are chosen as $\frac{k}{K}$ quantile of the data distribution. Subsequent data points are denotes as $x_{i,j}$, where $i = \{1,2,..,\mathrm{n}\}$. Observations in all intervals sum to the total number of observations, so $\sum_{k=1}^{K} n_j(k) = n$. Additionally let $k_j(x)$ indicates the range index number where a given observation $x$ falls, $z_{0,j}$ is set in close neighborhood to the smallest observation and just below it, whereas $z_{K,j}$ is equal to the largest $x_{i,j}$.

$$\overline{g_{ALE}}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{\{i:\, x_{i,j} \in N_j(k)\}} \left[\hat{f}\left(z_{k,j}, x_{i,\setminus j}\right) - \hat{f}\left(z_{k-1,j}, x_{i,\setminus j}\right)\right] \tag{8}$$

for each $x \in (z_{0,j}, z_{K,j}]$.

In (8) $x_{i,\setminus j}$ denotes the observation with subscript $i$ including all predictors but $j$. Figure 3 visualizes what ALE for feature $X_1$ does: after dividing predictor into $K = 5$ intervals, $\overline{g_{ALE}}(x)$ replaces the original feature value with the upper and lower interval values, calculates difference in predictions with replaced values within each of the interval, aggregates the results and centers. Equation (8) presents uncentered effect of feature $j$ on the predictions. Uncentered in the sense that it is accumulated across all observations. It is possible to obtain formula for centered local effect:

$$\overline{f_{ALE}}(x) = \overline{g_{ALE}}(x) - \frac{1}{n}\sum_{i=1}^{n} \overline{g_{ALE}}\left(x_{i,j}\right) = \overline{g_{ALE}}(x) - \frac{1}{n}\sum_{k=1}^{K} n_j(k)\overline{g_{ALE}}\left(z_{k,j}\right) \tag{9}$$

$\frac{1}{n}\sum_{i=1}^{n} \overline{g_{ALE}}\left(x_{i,j}\right)$ in the aforementioned equation accounts for average accumulated local effect for prediction, so the centered local effect measures how for a given predictor the prediction decreases or increases comparing to the prediction on average (Apley and Zhu, 2019). For example, let $X_1 = 10$ and $\overline{f_{ALE}}(X_1 = 10) = -3$. If predictor one is equal to 10, then the model outcome is lower by 3 with respect to the average prediction.

Creating partitions of data based on $\frac{k}{K}$ quantile of the data is easy for continuous variables, as these predictors often have some pre-defined order and dividing them into ranges is obvious. For categorical data, especially without exact hierarchy (for example marital status) computation of ALE effect of the predictor is possible, however it is necessary to somehow define the values order (Zhu et al., 2019). As ALE plot for categorical features computes the effect across some definite intervals also, for this reason it is necessary to define some hierarchy of the feature levels, so that the effect can be accumulated through the categories and transmit information about the model behavior consistent with the human logic.

### *2.3 Individual Conditional Expectation*

One more useful method designed in a similar manner as PDP, but for individual observations. It is designed for inspecting single observations of the model. Individual Conditional Expectation (ICE) curve is computed similarly as PDP, however the latter takes the effect of a given predictor for all observations and averages it. In other words, ICE requires plotting prediction as a function of $x_S$ conditional an observed $x_C$.

Remind model response function is $\hat{f}(\cdot)$. Consider observations set $\{(x_{i,S}, x_{i,C})\}_{i=1}^{n}$. For $n$ data points and a predictor $x_s$, $\overline{f_{ICE}}(x_{i,S})$ calculates predicted outcome of the model keeping $x_C$ fixed and changing the feature of interest $x_S$ ($x_c$ is different for all observations). As a result relation between prediction and predictor is shown for N observations. Taking average of the response results in Partial Dependence Plot.

Comparing ICE curve with PDP might bring us with interesting insights into trained algorithms. As Friedman (2001) noticed, Partial Dependence might be reliable only in case of weak dependence between predictors. Balancing these two methods and their results against each other can help detecting correlation between the independent features. It might be true especially where ICE curves for many instances are significantly different than average of them (Partial Dependence).

In terms of comparing the curves among many instances of the empirical data it might be desirable to set up centered ICE curve. It is helpful especially in a situation when observations start with different predictions and it is hard to capture differences among them. Centered ICE assigns a constant starting prediction for all instances (centers the observations) and then plots the curve with respect to the following equation (Goldstein et al., 2014).:

$$\overline{f}_{ICE.CENT}(x_{i,S}) = \overline{f}_{ICE}(x_{i,S}) - 1\ \overline{f}_{ICE}(x^{*}, x_{i,C}) \qquad (10)$$

where $i$ denotes the observation index, so $\overline{f}_{ICE.CENT}^{(i)}$ presents formula for single centered ICE curve. $1$ accounts for a vector of 1's and $x^{*}$is the anchor point, it might be either minimum or maximum value of $x_S$ or another value. Goldstein (2014) assures that in case $x^*$ is set to minimum of $x_S$, then all centered ICE curves are anchored at 0. This approach enables comparing predictions for single instances versus predictions for some fixed feature value.

### *2.4 Local Surrogate Models (LIME)*

This kind of explanation tool has been designed by for visualizing significant features of single observation in data. It is an agnostic tool, so it might be used for any kind of black-box model no matter how complex its structure is. Key concept of local surrogate models is fitting an interpretable white-box model around the instance of interest. For a linear model (such as linear regression) it is possible to derive a vector of $\beta$ coefficients for each of the model feature. LIME (Local Interpretable Model-agnostic Explanations) as an explanation toolbox incorporating concept of local surrogate models does it indeed by perturbing the data around the observations and estimating an interpretable model with the perturbed samples. LIME checks correctness of the prediction in terms of local fidelity and tests whether predictions of the model are locally faithful .

Mathematically, concept standing behind LIME computation is the following formula:

$$E(x) = arg\ \underset{g \in G}{argmin}\ L(f, g, \pi_x) + \Omega(g) \tag{11}$$

Aforementioned formula denoted as $E(x)$ stands for explanation of an observation $x$. Consider an interpretable model denoted here as $g \in G$, where $G$ is a class of explainable structures. It can be any model whose results might be clearly presented for a human, for example with the use of graph (such as decision tree) or whose features contribution can be computed as in linear models. $\Omega(g)$ is defined as a measure of complexity of model $g$. It is required that $\Omega(g)$ is small enough so that the $g$ is understandable for a human. Authors explain $\Omega(g)$ might point for the dept of the decision tree for example. With $f(\cdot)$ model prediction function is given. Function $f$ takes as an argument an observation $x$ and calculates prediction. $\pi_x(z)$ stands for a measure of distance between observation of interest $x$ and another instance $z$. $L(f, g, \pi_x)$ calculated precision of the explanation by comparing it to the original prediction with some measure, for example mean squared error. Algorithm LIME is responsible only for loss function $L$. Complexity measure is defined by a human.

LIME procedure involves the following steps: after picking an instance of interest $x$, LIME creates new, non-existing samples around the observation. After perturbations are made, predictions for these observations with black-box algorithm are calculated. Newly created samples are weighted by proximity measure $\pi_x$ and an interpretable model $g$ is fitted to them. Contribution of each predictor is calculated with respect to the result obtained with white-box model. Proximity measure is defined as an exponential kernel function as $\pi_x(z) = exp\ (\frac{-D(x,z)^2}{\sigma^2})$, where $D$ is a distance metric (for example Euclidean measure) and $\sigma$ is kernel width. Finally algorithm computes the following square loss function stated as:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \epsilon Z} \pi_x(z)\left(f(z) - g(z')\right)^2 \tag{12}$$

$Z$ is a set of perturbed around an instance $x$ observations.

Different implementations of LIME should allow for setting $K$- number of features that is used when explaining a prediction. When one considers higher number of predictors helps to explain the prediction better, it should be set with higher value. However in practice a few variables are enough. It is often possible to select different white-box model. Authors stress that it should not be random, as every interpretable model might have some drawbacks when explaining predictions of a black-box. They suggest using LASSO algorithm as it performs well comparing to other linear models (Ribeiro et al., 2016).

### *2.5 Shapley values*

Shapley values is a kind of unified approach designated for model inspection. With Shapley values numerous model agnostic global and local techniques for visualizing the model predictions might be obtained. This XAI tool is derived from game theory that studies interaction and actions of self-interested agents in a game. There is no exact definition of a game, however in general this is a set of rules that push the players to make their moves taking into consideration actions of the others. An utility function also exists. It rewards every agent with some utility (payoff) taking all players strategies as given (Hotz, 2016). In terms of explainable AI game theory can be considered as follows: for every of $k$ participants (predictors) the payoff is understood as prediction assigned to every possible combination of the players' move. It is assumed that agents have different strength in predicting the outcome For every possible combination of strategies (effects of the features) different payoffs are possible. Shapley values purpose is: for a given set of features and prediction calculated for them Shapley values calculate contribution of each player payoff to the final outcome.

Consider $n$ agents who represents predictors in the dataset. $S \subseteq N = \{1,2,\dots,n\}$ denotes the subset of agents and their number is set as $|S|$. $v(S)$ is the function assigning subset of agents $S$ with a total payoff contributed by co-working. For every possible game (prediction), according to the Shapley formula, $\phi$ is a function on the fixed set $N$ set and $\phi(v) = (a_1,\dots,a_n) \in R^n$. $a_i$ denotes the contribution of $i-th$ player to the final output $v(N)$

$$\phi_i(v) = \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!\,(|N-S|-1)!}{|N|!}[v(S+i)-v(S)], \qquad i = 1,\dots,n \tag{13}$$

where $S \subseteq N\setminus\{i\}$ is the subset of all but $i-th$ features from subset S and $v(S+i)$ is the function with the $i-th$ feature present. The function $\phi_i(v)$ computes in some sense marginal contribution of each player. It is done by summing the effect of each possible coalition of agents drawn from the $N\setminus\{i\}$ set. However it is done under assumption that all agents arrive randomly. In order to fairly distribute their power Shapley function averages the agents expected payoffs over $N!$ possible combinations. Another important assumption is that an empty set $v(\emptyset)$ is also valid in the calculations. Shapley assumes that $v(\emptyset) = 0$, as nothing is produced for free (Roth, 1988). In order to better capture the idea consider a set of two players $N = \{1,2\}$. There are possible four combinations consisting of the players; $\{\emptyset\},\{1\},\{2\},\{1,2\}$. Generally for subset consisting of $N$ players there are $2^N$ possible combinations. Shapley function for each of the $i = \{1,2\}$ player is given by:

$$\phi_1(v) = \frac{1}{2}[v(\{1,2\}) - v(\{2\})] + \frac{1}{2}[v(\{1\}) - v(\{\emptyset\})] \tag{14}$$

$$\phi_2(v) = \frac{1}{2}[v(\{1,2\}) - v(\{1\})] + \frac{1}{2}[v(\{2\}) - v(\{\emptyset\})]$$

Thus, Shapley value for each of the player is weighted sum of marginal contributions calculated as the difference of the payoff with and without a given feature.

Roth (1988) state that function $\phi(v)$ has some meaningful properties. The first one known as efficiency state that total output $v(N)$ is the sum of $N$ players contributions. In normal terms $v(N)$ would denote prediction for a single instance, however for the purpose of Shapley value calculations it is stated as model outcome minus average model prediction.

$$\sum_{i=0}^{N} \phi_i(v) = v(N) \tag{15}$$

Second property – symmetry - assumes that for a given subset of players $S$ and two different agents $i$ and $j$, so that

$$v(S+i) = v(S+j) \tag{16}$$

function $\phi$ is symmetric in the sense that it does not depend on the player name, only on its contribution to the function $v$. In other words, these two features contributions are equal (17).

$$\phi_i(v) = \phi_j(v) \tag{17}$$

Third axiom of dummy player – for any player $i$ if $v(S+i) = v(S)$ for all $S \subseteq N \backslash \{i\}$ then $\phi_i(v) = 0$. If contribution of feature $i$ does not influence the prediction, then its Shapley value $\phi_i(v)$ must be zero.

Lastly Shapley values satisfy additivity, Consider two games and two contribution functions $v$ and $w$. The gain from combining these two functions is equal to sum of individual gains for every player $i$:

$$\phi_i(v+w) = \phi_i(v) + \phi_i(w) \tag{18}$$

Additionally $\phi_i(cv) = c\phi_i(v)$, where $c \in R$. This last property is also known as linearity (Aas et.al., 2020, Štrumbelj and Kononenko, 2014).

Explanations standing behind Shapley values might be compared to the linear model, where prediction function is given by:

$$\hat{f}(x) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i \tag{19}$$

$\beta_0$ in equation (19) denotes the intercept, so average model outcome calculated over data distribution. Shapley values decompose predictions for single instances in a similar manner like linear models do, however the formers approach is designed so that it is applicable for every class of models. Let $x$ be an instance of interest, $\hat{f}(\cdot)$ prediction function and $\phi_0 = E[f(x)]$. Then prediction for the single data point is the sum of the expected outcome of the model and $k$ features contributions.

$$\hat{f}(x) = \phi_0 + \sum_{i=1}^{k} \phi_i \tag{20}$$

Actual idea behind presenting Shapley values for a given set of predictors is computing the difference between the feature effect and the average outcome. In order to calculate contribution of predictor $i$ to the score produced by the model for an observation $x$, it is necessary to derive formula for the difference when regressor value is not known. The difference in linear regression model (21) would be an equivalent to feature contribution obtained with Shapley approach.

$$\phi_i = \beta_i(x_i - E[X_i]) = \beta_i x_i - E[\beta_i X_i], \qquad i = 1, \dots, k \tag{21}$$

However, in practice no such explicit equation for non-linear models exists. In addition to that linear models often assume no serial correlation of independent variables. In order to overcome this problem equation (13) has been proposed and it is representation of Shapley value definition. Consider final output $v(N)$ that stands for prediction of a single data point. $|N|$ denotes number of features for that object. Shapley value for each of the predictor $i$ from the set $N$ is defined as $\phi_i(v)$, where $v$ is predictor value.

As computing expression $\phi_i(v)$ with $k$ number of features requires creating $2^k$ subsets and increases exponentially with the number of predictors, an alternative approach to (13) has been proposed by Strumbelj and Kononenko (2014). Its computation is based on Monte Carlo sampling. Consider an observation $x$, number of features in the model $k$, prediction function $\hat{f}(\cdot)$ and an integer number $m$ denoting the number of simulations,. Output of the simulation is contribution of a given feature $j$ to the final prediction. Strumbelj and Kononenko present the following algorithm for Shapley value of a single predictor:

for $1, \dots, m$:

- pick a random observation $w$ from the dataset
- pick a random permutation $O \epsilon \pi(n)$, where $\pi(n)$ is set of ordered permutations of the feature indexes
- create two new instances of the data $b_1$ and $b_2$ with the use of observations $x$ and $w$:
  - first order $x$ and $w$ feature values according to the sequence of indices in $O$:

  $x' = (x_1, x_{2,} \dots, x_j, \dots, x_{n)}$ and $w' = (w_1, w_{2,} \dots, w_j, \dots, w_{n)}$

  - cerate instances $b_1$and $b_2$ as follows:

  $b_1 = (x_1, x_{2,} \dots, x_j, w_{j+1}, \dots, w_{n)}$ and $b_2 = (x_1, x_{2,} \dots, w_j, w_{j+1}, \dots, w_{n)}$

  Instance $b_1$ is created by taking feature values of $x'$ for $i = 1, \dots, j$ and features values of $w'$ for $i = j + 1, \dots, n$. Analogously instance $b_2$ takes feature values of $x'$ for $i = 1, \dots, j - 1$ and features values of $w'$ for $i = j, \dots, n$.

  - calculate $\phi_j^m(x)$ of the m-th iteration from the formula: $\phi_j(x) = \hat{f}(b_1) - \hat{f}(b_2)$

Calculate average from the m iterations: $\phi_j(x) = \frac{\sum_{i=1}^{m} \phi_j^m(x)}{m}$. Shapley value for the feature $j$ .

This approach has been implemented in iml package in R. Another approach has been proposed by Staniak and Biecek (2018). Authors state their approach as an approximation of Shapley values. For a given observation $x$ it also computes contribution of each feature value to the final model score f̂(x). Its general idea is to start with an empty set, add all of the features one by one and calculate its highest contribution to the prediction. Features are added with respective to their predictive power. This model agnostic approach for explaining single predictions (Breakdown) has been presented in the algorithm below. It considers only step-up approach when calculations begin with an empty set and the features are added. Step-down approach also exist and requires calculating smallest distance between the full set of variables and the feature removed from it.

Once again consider a set of $k$ features and an observation $x$. Define $V$ as an empty set storing variable indices. Features indexes are added one by one depending on their contribution to the prediction. For the purpose of that it is necessary to define a function calculating distance between prediction for original instance $x$ and an instance with features from $V$.

$$d(x, V) = \left|\hat{f}(x) - \hat{f}\big(x(V)\big)\right| \tag{22}$$

$\hat{f}\big(x(V)\big)$ term measures prediction for $x$ with feature values fixed on variables from $V$. For the rest of the features it is assumed they follow the population distribution

```
$V \leftarrow \emptyset$
for $i$ in $1, \dots, k$:
    for $j$ in $1, \dots, k \quad V$
    ▪ for the observation $x$ calculate the distance with the feature $j$ added to set $V$ –
          $d(x, V + j) = \left|\hat{f}(x) - \hat{f}\big(x(V + j)\big)\right|$
    ▪ choose the feature that maximizes the distance and denote is as $j_{max}$
    ▪ calculate contribution of the feature $\phi_i(x)$ as difference of prediction made with the
      feature $j_{max}$ and without it:
      $\phi_i(x) = \hat{f}(V + j_{max}) - \hat{f}(V)$
    ▪ add the feature $j_{max}$ to the $V$
```

Variable contributions in this procedure are ordered with their decreasing predictive power. Algorithm chooses the most powerful feature at first and then adds iteratively less important predictors (Staniak and Biecek, 2018). The main drawback of this method is that contribution of consecutive variable is dependent upon previously added predictors. This result might be upset in case of strongly correlated predictors.

Shapley values used for the purpose of local model explanations are very similar in their destination to local surrogate models implemented in LIME. By contrast to surrogate models that assume linearity of the classifiers, the concept deriving from coalitional game theory has strong mathematical background. Efficiency, symmetry and additivity axioms assure that for each data points features contributions are distributed in a “fair way”. Feature contributions for all regressors must be distributed so that the sum of effects and the average adds up to the local prediction. Moreover, if a feature value is not present among regressors, its Shapley value is surely 0. Techniques incorporating Shapley values are not only powerful and reliable tools for machine leaning local model inspection, but can be used from different perspectives. Equation

(13) serves for local explanations, but by summing features contributions for all of the instances and averaging the effect, obtaining marginal contribution is possible. Visualizing the Shapley values with respect to a chosen predictor values and obtaining partial dependence plot is also possible. One has to bear in mind that once calculating feature contribution $\phi_i$, all explanatory variables are taken into consideration. Explanations produced by local surrogate models often allow to specify number of predictors for the user. Comparing to LIME procedure, estimating (13) for k predictors requires $2^k$ combinations of payoffs. Calculations of feature effect would be time-consuming even for one observation, not mentioning the whole dataset. There are some concerns regarding sampling feature values when calculating payoffs with and without the player. In order to predict the model outcome, missing regressor must be randomly sampled from the data. Slack et al. (2020) raise the issue that both LIME and Shapley values computed in different packages (such as SHAP in Python) are perturbation-based methods whose results are always biased in some way. However, in order to overcome this problem, m denoting number of Monte Carlo simulations in Štrumbelj and Kononenko (2013) algorithm might be set at sufficiently high level, what unluckily increases computational time.

## 3. Dataset

Assessment of the explainable tools for Artificial Intelligence has been performed for credit bank data from Kaggle – online community of data scientists, analysts and machine learning competitors. The data consists of past clients history and information such as length of employment, age, purpose of the loan, duration of the loan etc. Dependent variable – default – states whether the client managed to fulfill its obligations to the bank. There is no clear definition of the default given with this dataset. These definitions might be different under many regulators and as it is not the main purpose of this paper, further investigations are not made. Table 1 summarizes a total of 20 independent variables plus the binary dependent variable indicating whether a customer defaulted or no. There are no missing values in the data so handling the problem is not present here. The only transformation was converting non-numeric categorical features into numerical and creating dummies for categorical features with no order or hierarchy among their unique values. It is decided that checking balance, credit history, savings balance and employment length regressors will be fed into the learning algorithm as converted numerical features with pre-defined ordering. Hierarchy of the values is equivalent to the ordering from the details column in the table. For example, checking balance unknown is set to 0, < 0 DM is equal to 1, 1 – 200 DM – 2 etc. Credit history critical is set to 0 and so on. Rest of the categorical input drivers are transformed to dummies. After feature encoding, there are 41 explanatory variables in total.

**Table 1. Description of explanatory variables fed into the random forest classifier. Label is the original feature name accepted by the algorithm and plotted in the explainable tools. If type of predictor is categorical, the feature is encoded as numerical or the dummy variable is created accordingly. Detailed description of features categories is included in last column.**

| Label | Description | Type of variable | Variable details |
|---|---|---|---|
| checking_balance | Amount that the borrower owes to the bank. | Categorical | unknown, < 0 DM, 1 – 200 DM, > 200 DM |
| months_loan_duration | Duration of the loan in months | Numerical | |
| credit_history | Payment history informing how timely one repays his debt. | Categorical | critical, delayed, repaid, |
| purpose | Purpose of the loan. | Categorical | radio/tv, car (new), furniture, car (used), business, education, repairs, others, domestic appliances, retraining |
| amount | Amount of the loan granted to the client. | Numerical | |
| savings_balance | Savings balance. | Categorical | unknown, < 100 DM, 101 – 500 DM, 501 – 1000 DM, > 1000 DM |
| employment_length | Years of employment. | Categorical | unemployed, 0-1 yrs, 1-4 yrs, 4-7 yrs, > 7 yrs |
| installment_rate | Installment rate. | Numerical | |
| personal_status | Marital status. | Categorical | single male, female, married male, divorced male |
| other_debtors | Debtor of a loan | Categorical | none, guarantor, co-applicant |
| residence_history | Residence history of a borrower. | Numerical | |
| property | Property owned by a borrower. | Categorical | unknown/none, other, real estate, building society savings |
| age | Borrower's age. | Numerical | |
| installment_plan | Installment plan. | Categorical | none, bank, stores |
| housing | Type of housing. | Categorical | own, rent, for free |
| existing_credits | How many existing credits a borrower has. | Numerical | |
| **default** | Dependent variable – whether a borrower has defaulted. | Categorical | 0, 1 |
| dependents | Number of people relying on borrower's for support. | Numerical | |
| telephone | Whether borrower's telephone number is known. | Categorical | yes, no |
| foreign_worker | Whether borrower is a foreign worker. | Categorical | yes, no |
| job | Categories defining employment of a borrower. | Categorical | skilled employee, management self-employed, unskilled resident, skilled employee |

*Source*: own preparation.

For the purpose of presenting aforementioned tools, training a machine learning model is required. Since most of the explainable tools are model agnostic (designed for all kinds of models), random forest has been chosen as an example. Algorithm has been built in terms of classification problem, as only two possible outcomes (default or non-default) are possible Random forest parameters have been chosen with the grid search method. Tuning parameters of an estimator is done with 5-fold cross-validation in order to improve performance of the algorithm and preserve its stability. Since random forests belong to tree-based family of models that are resistant to feature scaling, no standard scaling or normalization of the features has been performed. Different combinations of the following parameters have been checked: number of estimators in the random forest, maximum number of features used in training has been set to 'auto' or to squared value of number of features. Maximum depth of the trees, minimum sample split, minimum sample leaf and bootstrap parameters have been also searched for best values. Both the model and XAI methods have been performed with Python and its open-source packages. Finally model with the following parameters has been chosen as the best: maximum depth was set to 30, maximum number of features is automatically chosen by the random forest object, minimum sample leaf is 1, minimum sample split is 7 and overall number of estimators in the random forest is equal to 1200. For the purpose of validating the model on the independent test set not used for the model learning, before diving into parameters tuning the whole dataset has been split into training and test set. The former one is separated for the purpose of model learning process, and the latter sample is used with the fully-specified classifier for the evaluation only. Parameters are searched with respect to the training sample and criterion measuring quality of the split is Gini Impurity.

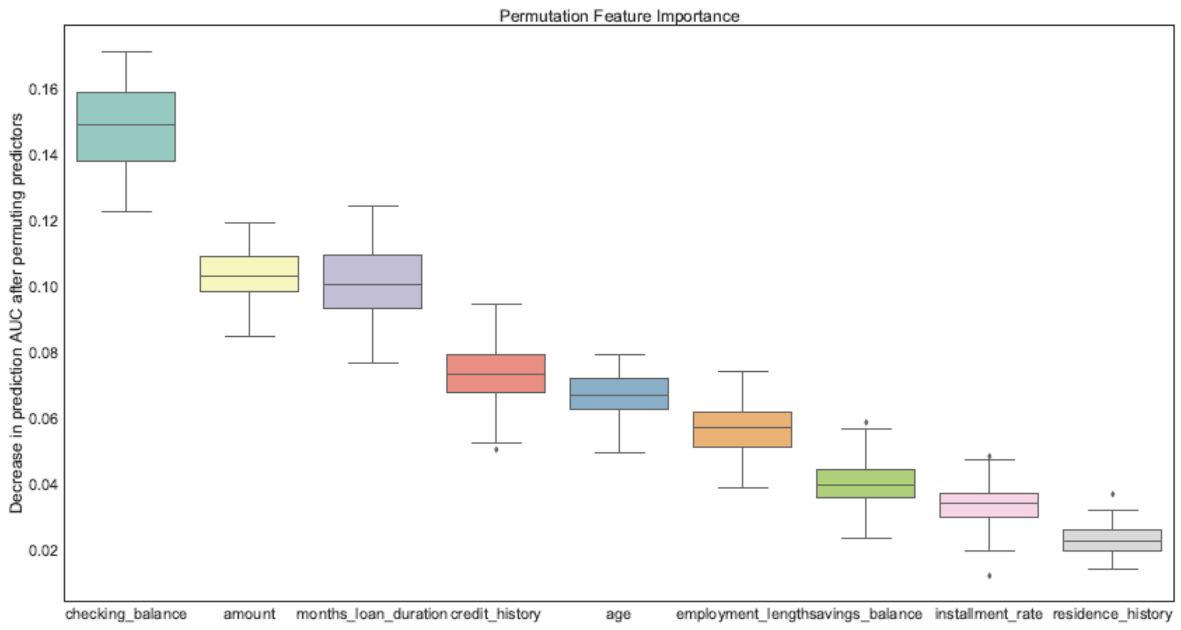
## 4. Results

### *4.1.Permutation Feature Importance*

It is often desirable to see what predictors account for the biggest part of variability of the data. Feature importance with permutation approach is a solution for that. Multiple libraries enable computing Permutation Feature Importance in Python. As random forest model has been performed with scikit-learn, obtaining feature importances straight from the trained object is possible. It should be noted that these values aren't evaluated by permuting the feature values. Each random forest consists of many single decision trees. In order to split the variable so that it optimally discriminates two distinct sub-groups within a dependent feature, usually Gini index or Information Gain and entropy is used. It serves as a impurity measure. After split of the variable into partition, reduction in Gini index (entropy) is calculated. Feature importance in random forest algorithm has been primary calculated by Breiman (1984) and employs these impurity gains. It is evaluated by computing how much the impurity decreases for all of the nodes and taking the average from all of the trees in the forest. Impurity decreases in single trees are weighted by ratio of samples reaching a distinct node to the total number of observations in the sample. This variable importance calculation is often referred as Mean Decrease Impurity importance. Subsequently, Mean Decrease Accuracy (MDA) has been proposed by Breiman (2001), that incorporates permutation. Feature values are randomly shuffled in the out-of-bag data and decrease in accuracy measure is computed (Louppe et al., 2013).

Instead of Mean Decrease Impurity importance obtained from the trained random forest object, one can simply develop features ranking by permuting each of the dataset variable and measure how much this action influences a chosen metric. In order to get more reliable results this procedure should be repeated many times in order to average the random component from the trials. This simple simulation has been performed independently using AUC as a measure serving for importance ranking. Instead of AUC any metric could be defined as a scoring function. In case of a classification problem this metric must accept categorical outcomes of the model.

**Figure 4. Permutation Feature Importance results obtained with random forest classifier. Figure depicts scores for own implementation of PFI where AUC (Area Under the ROC Curve) is chosen as performance metric. Vertical axis is amount by which prediction error increases when randomly shuffling each predictor values. Weights for the method are obtained by randomly shuffling predictors 100 times.**



*Source*: own preparation.

Results of the computations have been visualized for nine most influential variables in the boxplots in order to show distribution of changes in prediction AUC. On the vertical scale magnitude of predictor influence for prediction scores is located. Horizontal scale positions attributes according to their feature importance decreasing, so in this case checking balance is the most influential feature. In other words, after shuffling checking balance values one hundred times, calculating decrease in prediction AUC score and averaging them, this feature turns out to be the best in terms of discriminating the defaults and non-defaults. It seems to be reasonable as checking balance is highly correlated with probability of default. Highest value for prediction AUC decrease for this variable is as high as about 0.15. It can be understood that decision model depends on checking balance in about 15%, which is very high. Second most relevant predictor is amount of the loan granted to the client, however score for months loan duration is comparable. The former variable exhibits lower variability of the scores. For both features median effect of permuting their values is about 10%. Rounded decrease in prediction AUC for these three regressors is as many as 25% of total model AUC. Next important variable is credit history of the credit recipient. It is sensible as many banks derive knowledge about credit history of the clients from institutions gathering knowledge about credit obligations and loan repayments history. Mihai et al. (2018) confirm that credit history is one of the crucial drivers of probability of default. Considering validity of age in the rank of most influential attributed it is confirmed by the literature that borrowers age and probability of default is strongly correlated. For example Debbaut, Ghent, and Kudlyak (2013) confirm this hypothesis. Moreover they find that risk of falling into default for middle-aged borrowers is highest. The youngest and elderly people, on the other hand, manage to fulfill obligations on time. Other important variables within learning set are: employment lengths, savings balance, installment rate and residence history. Polena and Regner (2018) proved significance of employment length in predicting borrower default, however Mihai et al. (2018) confirm current occupational status to be significantly correlated with the probability of default for both mortgage and consumer loans. Most of the predictors obtained with PFI method seem to be consistent with human logic and the literature.

When implementing this model explanation tool, it should be remembered that results might vary greatly for each single simulation. However the higher the number of simulation, the tighter error bounds are. Spread of the AUC prediction changes is represented in the boxplots, where distance between highest and lowest result as well as median is computed. It should be noted that an important feature for the system might be assigned with low average permutation score because of a small outlier and so reverse is true – irrelevant feature in practice might be become significant because of the outlier with high PFI value (Huang et al., 2016). For this reason plots visualizing distribution of calculations are recommended. Another factor influencing the results is also a scoring function. Ranks obtained with accuracy would be undoubtedly different as they only measure correctness of classification for a given cutoff.

After computing contribution of each predictor to the overall prediction AUC decrease, it is verified that nine most valuable features account for about 67% of total prediction error increase after permuting. Yet another interesting conclusion arriving from this method is that type of predictors considered as significant matters here. Numerical features are those giving the model most insights about the clients default. In other words, they have highest predictive power. Whether it is true in other cases, it should be tested empirically for sure, however in case of predicting clients default features such as borrower's age, loan's amount or duration of the loan seems to be reasonable in defining whether a client meets his obligations on time.

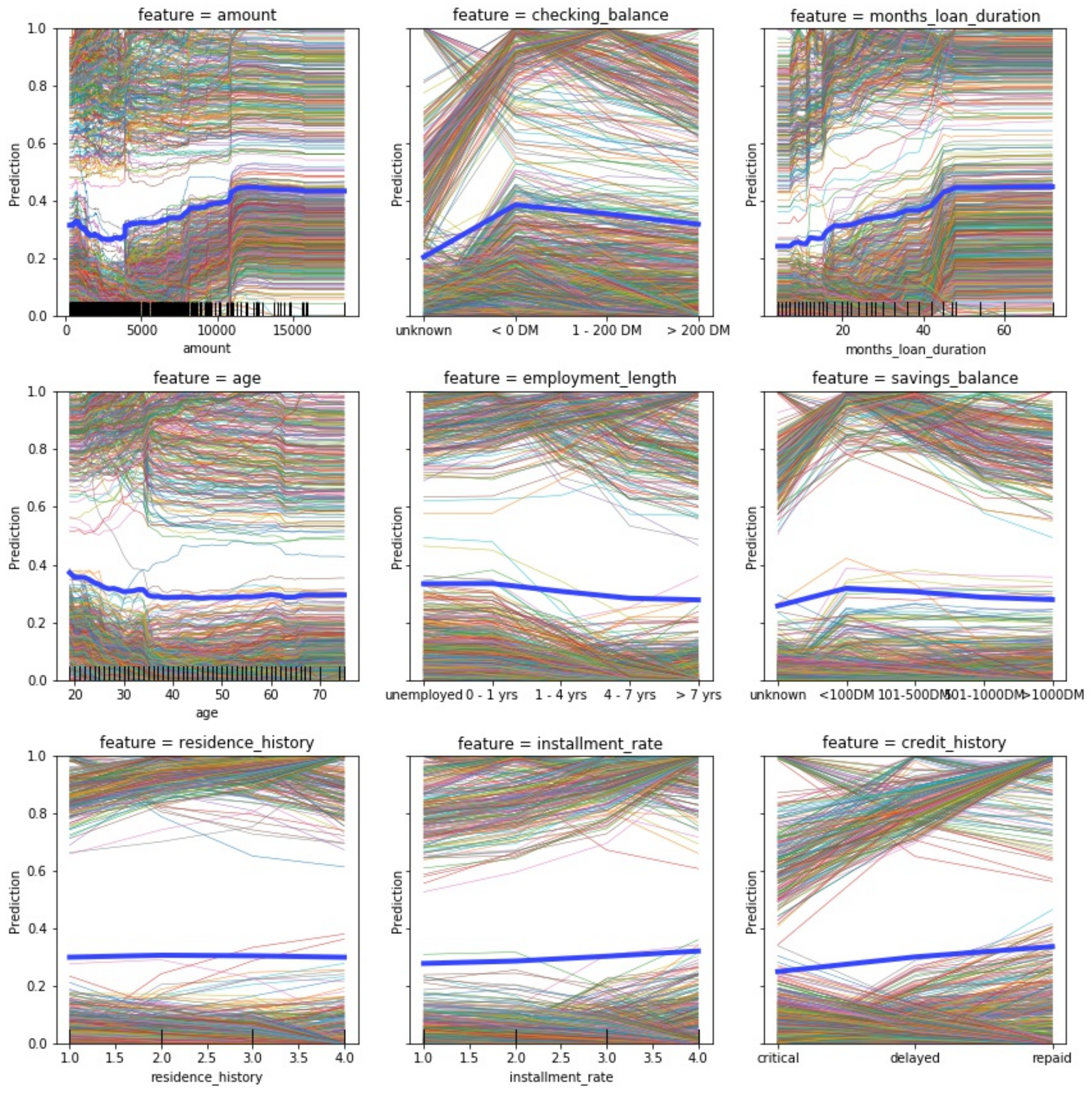
Feature importance score informs how much prediction error increases when individual predictor is perturbed. If the score is high, input variable contribution is also substantial. This tool contributes first-hand knowledge about "scoring behavior" of the model in the global terms. If there is post-hoc knowledge about serious reliance on that feature, one could ascertain the same feature is essential in explaining a certain event, such as credit default. There could be a case, when a random variable without no information value turns out to be meaningful according to PFI. This shall be a case when PFI is a mark of some bias in the model. Feature importance can be valuable not only when testing significance of regressors, but also with checking stability of the black-box model. It can be done by plotting mean decrease in prediction for the features in two samples – training set used for model estimation and parameters tuning and test set. The latter one should serve for evaluation purposes only. As machine learning models tend to overfit to the learning samples and thus, their evaluation metrics with independent datasets are usually smaller, it is obvious that PFI scores obtained with the two samples would be usually different. Nevertheless, in order that the model explanations could be perceived as stable, PFI scores should be comparable when computed with different samples. At least, feature ranks should be maintained.

### *4.1.Partial Dependence Plot*

In this section plots revisiting model outcome sensitivity with respect to the explanatory variables grid values are presented. Only a couple of predictors will be plotted as the space is limited. Moreover it is believed that analyzing relationship between the outcome and most significant variables in the model is desirable, as these predictors have highest impact on predictors.

In this section plots drawn with the use of PyCEbox Python package are presented. This module enables easy development of PDP as well as ICE (Figure 5) and centered ICE (Figure 6). Apart from plotting relationship between predictions of individual observations versus predictors' values in a standard way, PyCEbox makes it possible to choose a set of unique grid points of a predictor for the calculations. In the paper all of the features' values are incorporated into visualizations. Apart from that, it is possible to specify whether original data points should be plotted. Presented graph illustrates ICE curves and Partial Dependence at once. As the package exploits original data points from the model training process (Python requires that all features are numeric), plotting ICE curves with PyCEbox would show relationships for encoded labels. In order to present original feature values (instead of encoded categorical values into numeric) slight modifications to the function plotting the ICE and PDP values have been done. Each line with different color points at different observation. Partial Dependence is marked with bold blue line.

**Figure 5. Individual Conditional Expectation curves and Partial Dependence visualizations of predicted probability of default for nine most influential explanatory variables according to PFI. Thin colorful lines represent ICE, while bold blue accounts for Partial Dependence.**

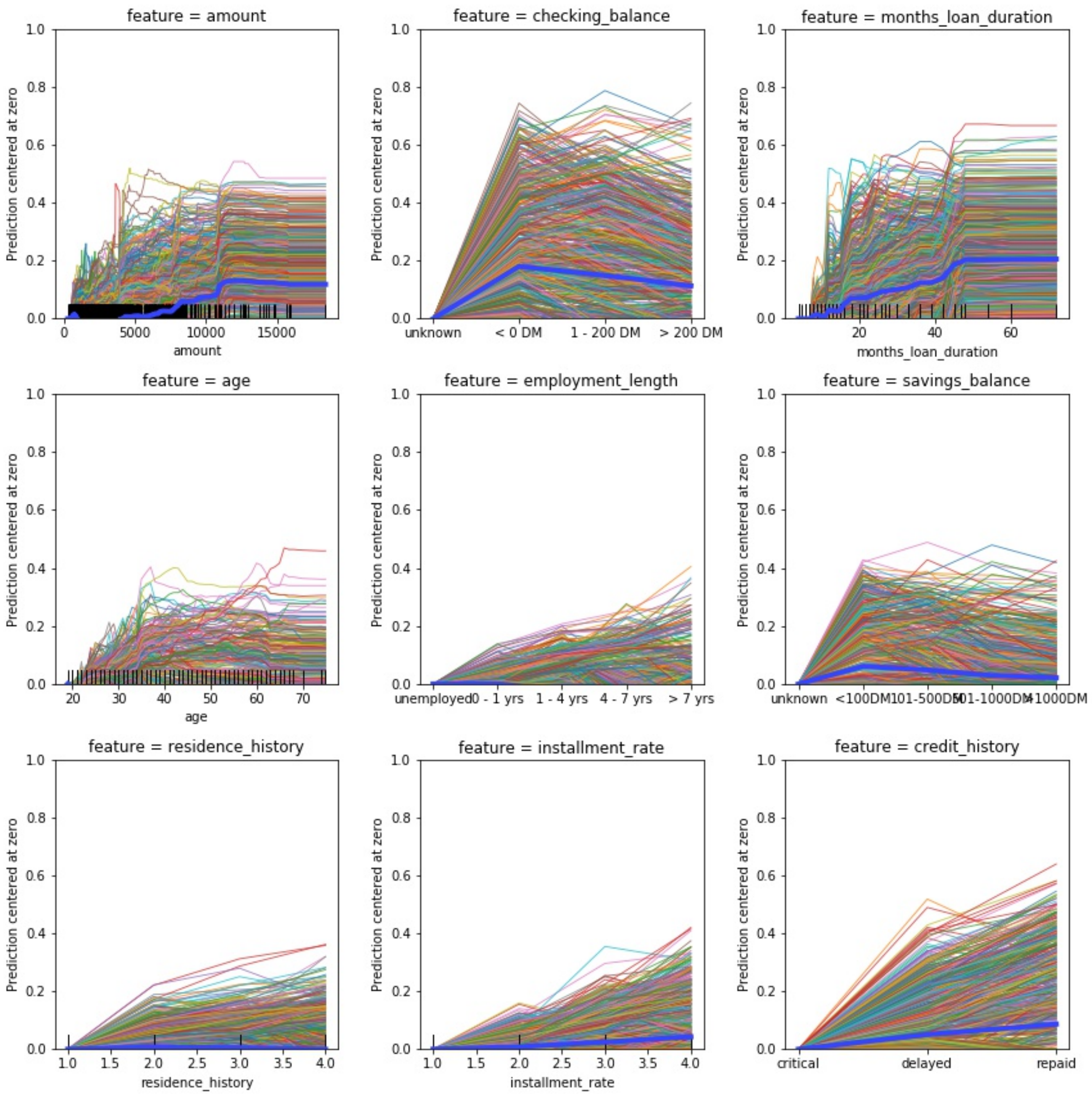


*Source*: plots obtained with PyCEbox Python package.

From the Partial Dependence some pretty conclusions might be drawn. There are some features for which ICE curves and Partial Dependence are rather constant for all feature values. In other words, predictions for the instances do not change a lot when different values of the variable are assigned to the observation. It is true for installment rate and residence history for example. No matter what the predictor value is, we would still assign the same binary result (1 or 0) for most clients, taking into account predefined cut-off. However amount, checking balance or loan duration in months strongly influence predictions for presented instances not only in terms of local predictions but also for the whole analyzed set of observations. In terms of duration of the loan shape of ICE curves might indicate that granting long-term loans results in higher probability of default for many cases. Considering savings balance probability of default increases for many instances when the balance is <100 DM. Once savings obtain higher rate probability of default decreases. The same is true with employment length. More risky clients (those with high probability of default) obtain lower score with higher job seniority. Consistent with the intuition properties are visible for Partial Dependence of age (relation between its shape and predictor values). Average prediction slowly decreases as the clients become older. This might be explained as lower riskiness of people with higher age comparing to younger and often unexperienced workers. The only relationship visible in the plots and contrasting with human logic is PD for credit history. In the figure it is visible that average score is higher when the client has repaid its obligation in the past, and lower when he is late with payments. It should be considered whether the feature categories have been encoded properly. Undoubtedly PD and ICE curves for that feature reveal some space where the model cannot be trusted and its reasoning should be improved.

Perhaps more diversity among predictions can be seen in curves centered around some constant value. Next graph visualizes this approach with prediction centered around zero. This approach enables comparing predictions among all of the instances. What centered ICE curves show is that predictions for different length of employment do not increase a lot comparing to the centered value. They are almost stacked at each other. In terms of loan amount granted to the clients it is problematic to notice heterogeneity in the data in uncentered curves as the average prediction for the clients is more and less the same. Clear visible patterns in predictions are visible in the centered curve. It is observed that compared to amount 0 predictions remain unchanged till amount is less than 5000 and then probability of default increases. Cumulative effect of different checking balance levels is as follows: maximum average prediction is highest comparing to unknown value of that feature when balance is “<0 DM”. Once level of checking balance increases, prediction falls but still remains higher than in the initial unknown state. It should be considered whether the model reasoning is correct. Answering the question whether checking balance unknown value is indeed better than “<0 DM”, it’s the matter of data. Since no detailed description of the feature levels have been provided it is not within the scope of this paper to analyze the addressed question.

**Figure 6. Centered Individual Conditional Expectation curves and Centered Partial Dependence visualizations of predicted probability of default for nine most influential explanatory variables according to own implementation of PFI. Thin colorful lines represent ICE, while bold blue accounts for Partial Dependence. All curves are fixed to 0 at the minimum feature values.**

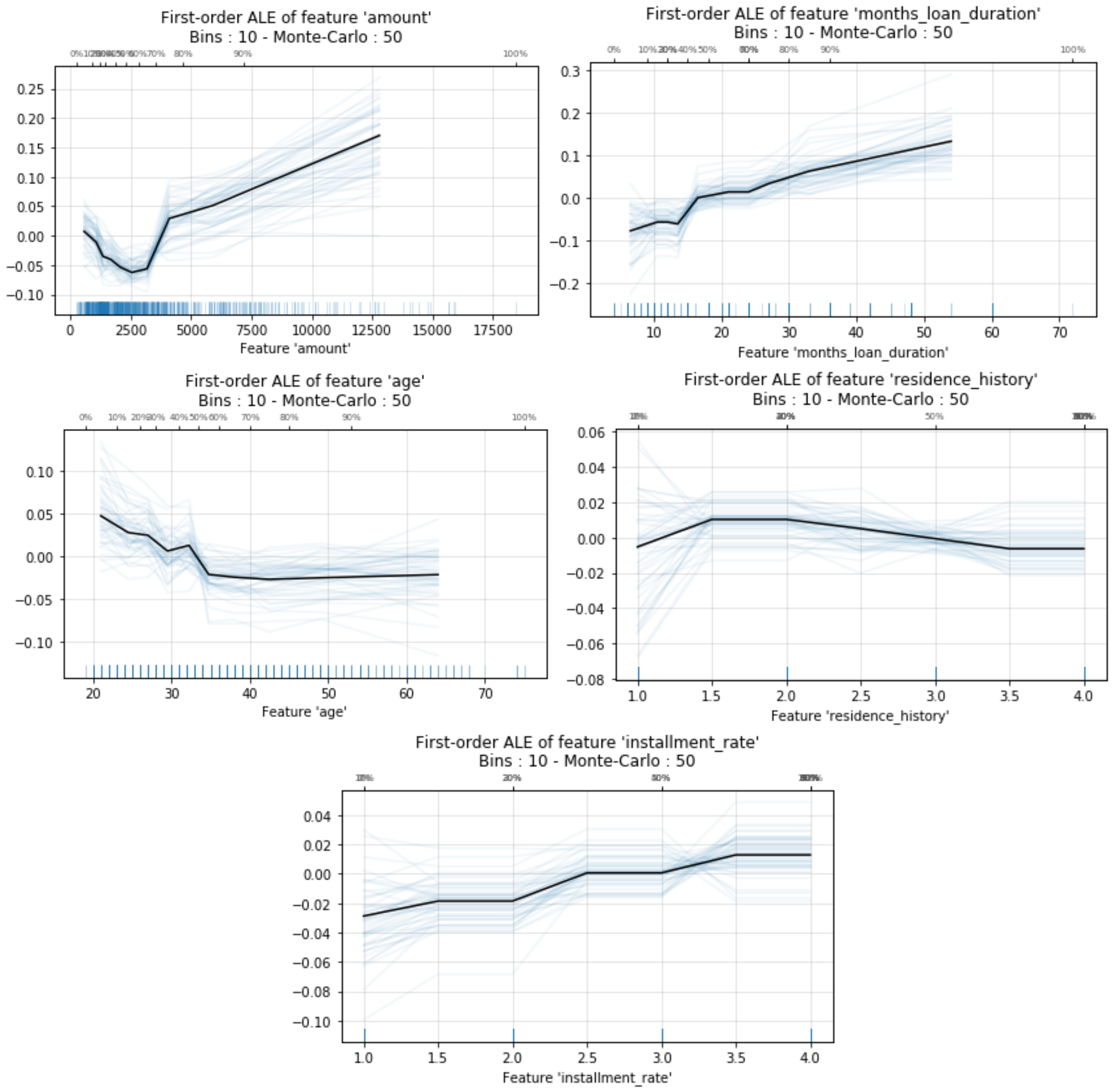


*Source*: plots obtained with PyCEbox Python package.

Partial Dependence Plots and Individual Conditional Expectation Curves reveal knowledge about discriminant power of explanatory variables. They should be produced once relevant features in the model are detected. Plots revisiting model outcome with respect to changing input values is crucial in terms of credit scoring. ICE shows how probability of default for a definite customer changes when the customer applies for a higher loan. PD shows the change in the whole sample. Visualizing average response of the learning structures by incorporating marginal distribution has some weaknesses. If collinearity among the variables exists, relationships captured by these tools will be upset by the interactions. Suppose trust in the model is achieved because accuracy of the model is high and model performance is stable over variations in data. In such a case many would be misled that marginal effect detected by these methods is credible. In addition to that nonexistent data points are created when assigning a custom feature value to all observations. For the Partial Dependence of dependents (e.g. dependents=3) averaging is done over all possible observations. Take for example a single unemployed male, taking long for the education purposed. It is simply impossible to assign him with three dependents, as supposedly he is a poor student maintained by his parents. Although these two methods are very intuitive and their interpretation is easy, they carry potential risks in correct understanding credit default forecasting. PDP and ICE simply evaluate misleading information in case serious correlation in the data is detected.  An alternative to these methods is Accumulated Local Effect implemented in ALEPython package, that mitigates biases contributed by its predecessors

ALEPython, as the author assures, deals with the problem of highly correlated features, as instead of marginal distribution, it takes into account conditional density of the variables. Apart from number of bins (intervals) used to split the feature' values, it is also possible to select number of simulations in Monte Carlo. It is so as ALE plots are computed with Monte Carlo method. Proportion of randomly selected samples at each Monte Carlo iteration can be also set. Monte Carlo samples are shown with blue lines in the figures. The black line indicates what is the average value believed as the true impact of the feature on the prediction. The higher the number of Monte Carlo iterations, the more accurate are the estimations. Accumulated effect centered around the mean prediction is presented. It is believed that the shape of PDP and ALE curves do not differ significantly, so serial correlation within predictors exists in the data. Author of the package stresses, that first-order (presented in this paper ) and second-order ALE plots have been developed only. First-order and second-order ALE plots for categorical features are still in progress. Thus, in this paper only numerical continuous features have been incorporated into development of plots with the use of ALEPython. For categorical features from the range of most significant predictors (determined with PFI scores) own function calculating accumulated local effect across the categories have been developed. Centered effects showing how for a given predictor the prediction decreases or increases comparing to the average prediction have been visualized in the bar plots.

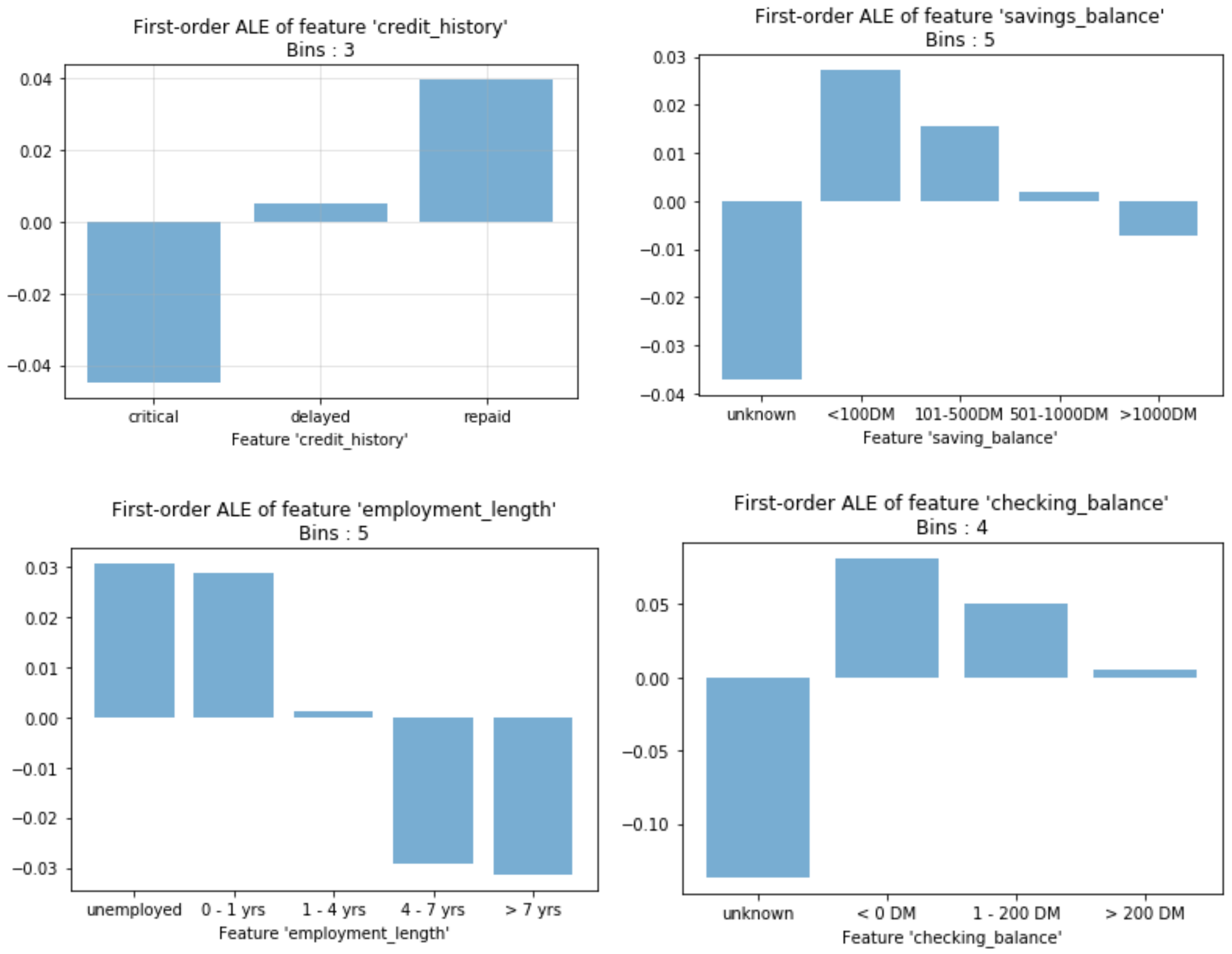
**Figure 7. Accumulated Local Effects visualizations of predicted probability of default for five most influential numerical explanatory variables according to PFI. ALE are centered – at each point prediction minus average model outcome is shown. Thin blue lines represent Monte Carlo sampling results, while bold black line is estimated true predictor influence equal to the average value of Monte Carlo iterations. Grid lines in the bottom of ALE plots show the distribution of feature values.**



*Source*: plots obtained with ALEPython Python package.

Comparing the shape of the curves (for continuous variables) estimated with partial dependence and accumulated local effects, it seems that the former do not inflate significantly the average impact of regressors on probability of default. At least it estimates direction of predictions moves (increase or decrease) properly. Of course, the latter better reflects the pure effect of individual feature values on the outcome as the curves are not monotonic. PD curves are rather straight and do not reveal much asymmetry between observations when predictors change by a small value. ALE on the other hand make the changes visible even for small movements of the predictors. Partial Dependence Plots are biased for almost all predictors at least locally, where nonexistent datapoints inflate the data distribution. It is clear that amount, months loan duration and checking balance influence the model outcome strongly. For the residence history ALE capture nonlinearity relationship between the model outcomes and residence history or installment rate. For the amount variable PDP is biased by the observations assigned with high loans. Because number of clients with loans exceeding 10000 is low and PDP does not correct for that fact, ALE estimates are more accurate. Yet another interesting relation that was not captured even by individual conditional curves but the arises from the ALE plot is impact of employment length on the probability score. It correctly reflects the fact that the average prediction rises when the client is unemployed or his working history is short. Probability score decreases for longer job seniority. Still influence of credit history is surprising. Undoubtedly this should be corrected in the model.

**Figure 8. Accumulated Local Effects visualizations of predicted probability of default for four most influential categorical explanatory variables according to scikit-learn PFI. ALE are centered – at each point prediction minus average model outcome is shown. Number of bins is set to unique values of predictor levels.**



*Source*: own preparation.

Suppose trust into estimated random forest is built and demonstrated ALE plots reveal how explanations withing predictors are generated. With this conceptual framework more insight about behavior of the risky clients can be deducted. Some hypothesis stated before modelling can be confirmed or rejected. Many discriminant patterns separating good and bad clients (bad are those delayed with loan repayment) can be verified. Discriminant power of the predictor is visible especially in case when Accumulated Local Effect curve is not flat, but for example monotonic and increasing. Slope of the months loan duration ALE curve could be approximated with a positive constant revealing that with increase of duration of a loan, the client becomes more risky. ALE method can be successfully adopted for explanatory purposes of the credit scoring model.

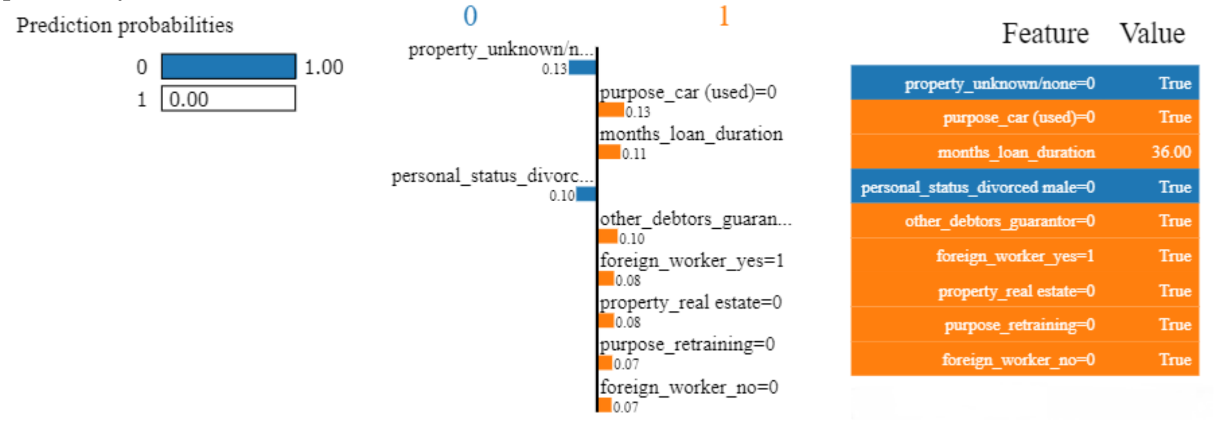
## *4.2.Local Surrogate Models*

Machine learning models are known with their predictive power superior to simple statistical structures. Main reason standing behind is their inner complex structure and no pre-defined assumptions about data density. As ML models usually succeed in high accuracy and evaluation metrics, it deceives the users into trusting them and believing black-box structures as correct and fair. Relying on the models in terms of accuracy of predictions is major, however once it is checked how the model predicts the outcome (if it predicts based on significant in our opinion patterns) it can be proved whether the algorithm can be trustworthy or not. Local model explanations have been developed in order to prevent misguided belief that the black-box structures can be always trusted.

Key idea of local surrogate models is to fit an interpretable model in close vicinity of an instance of interest and explain what features decide about the model score. LIME is probably the most popular package exploited for these purposes. It a model agnostic tool for explaining single predictions of any classifier or regressor (Ribeiro et al., 2016). It supports both tabular and text data, as well as deals with image recognition problems. LIME accepts classifiers, regressors and handles multiclassification problems. As LIME perturbs instances locally around the point of interest, Python implementation of this toolbox enables setting the parameter defining number of samples to be created. For numerical features observations are taken from the standard normal distribution and inverse standardization is done, however for categorical predictors LIME samples from the original distribution. There is also parameter defining number of features to be used in the explanation. The higher number, the more accurate the explanation will be. Apart from adjusting number of features for the explanation, kernel width in the exponential kernel defining proximity measure can be manipulated. The higher the width, the better LIME fits locally to the data. For example, consider LIME explainer applied to our random forest classifier on the training set with kernel width equal to 5. Predictions produced by the explainer bring us with R-squared of about 0.2. On the other hand, decreasing the parameter to 2 increases R-squared to 0.98. Defining own kernel function that incorporates the distance measure is also possible.

Below example of LIME implementation is generated for an instance from the independent data set (testing sample) with high prediction error. We want to check what features decided of assigning this observation with a very low score, even though the client defaulted in reality. Features and values in the table show the actual client features. If the value is set to True or False, the feature is encoded as the dummy variable. Those predictors marked with blue  contribute negatively to the higher score (decrease probability of default), whereas orange feature values bring the observations with higher probability. Final prediction of the random forest equal to 0 (credit recipient is considered to be a nonhazardous person), however score produced by the local surrogate is usually different (in this case it is 0.24). This is the main drawback of this method comparing to Shapley values, whose properties assure that adding all features contributions and average model outcome produces final model score. Float numbers in the middle part are the weights assigned to each of the nine most important to LIME features. In the following case whether property is unknown or none for the surrogate is model most significant. The second predictor is negation of taking loan for the purchase of a used car. Third most influential is months_loan_duration variable. Its lengths of 36 months pushes the score higher. On the other hand, fact that credit recipient is not divorced male makes probability of default lower. Other important attributes values contributing positively are other_debtors_guarantor, foreign_worker_yes, property_real_estate, purpose_retraining, foreign_worker_no.

**Figure 9. Local surrogate model (LIME) visualization of nine most influential in scoring features for an observation from the test set with high prediction error. Features in orange have positive and features in blue negative impact on increasing probability of default. Model predicted outcome, so calculated probability of default is 0.**

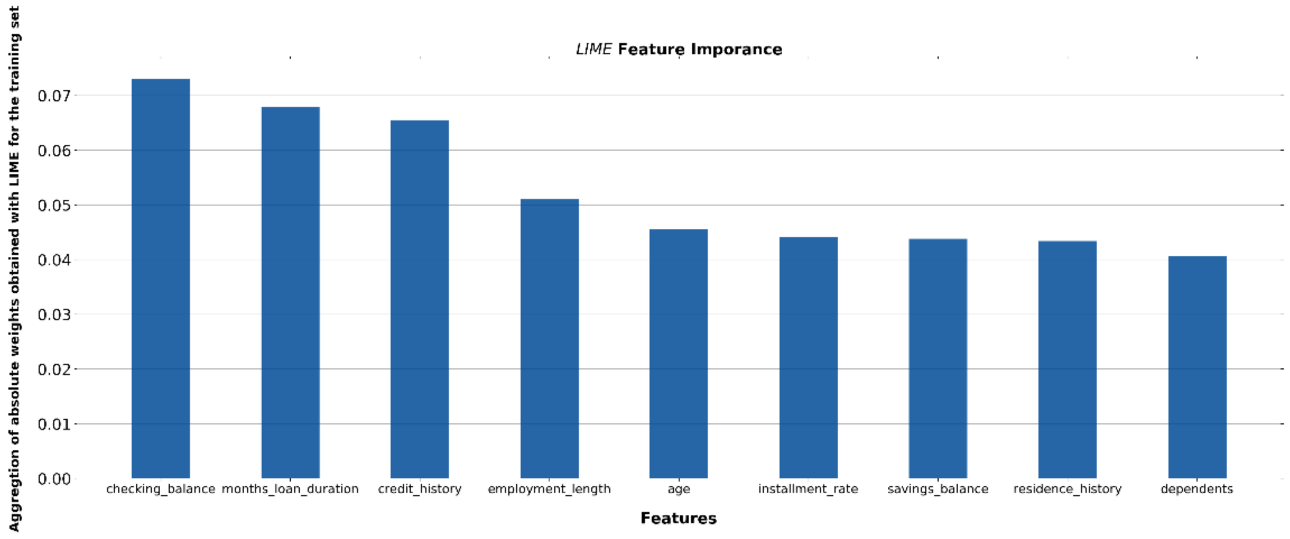


| Feature | Value |
|---|---|
| property_unknown/none=0 | True |
| purpose_car (used)=0 | True |
| months_loan_duration | 36.00 |
| personal_status_divorced male=0 | True |
| other_debtors_guarantor=0 | True |
| foreign_worker_yes=1 | True |
| property_real estate=0 | True |
| purpose_retraining=0 | True |
| foreign_worker_no=0 | True |

*Source*: plot obtained with LIME Python package.

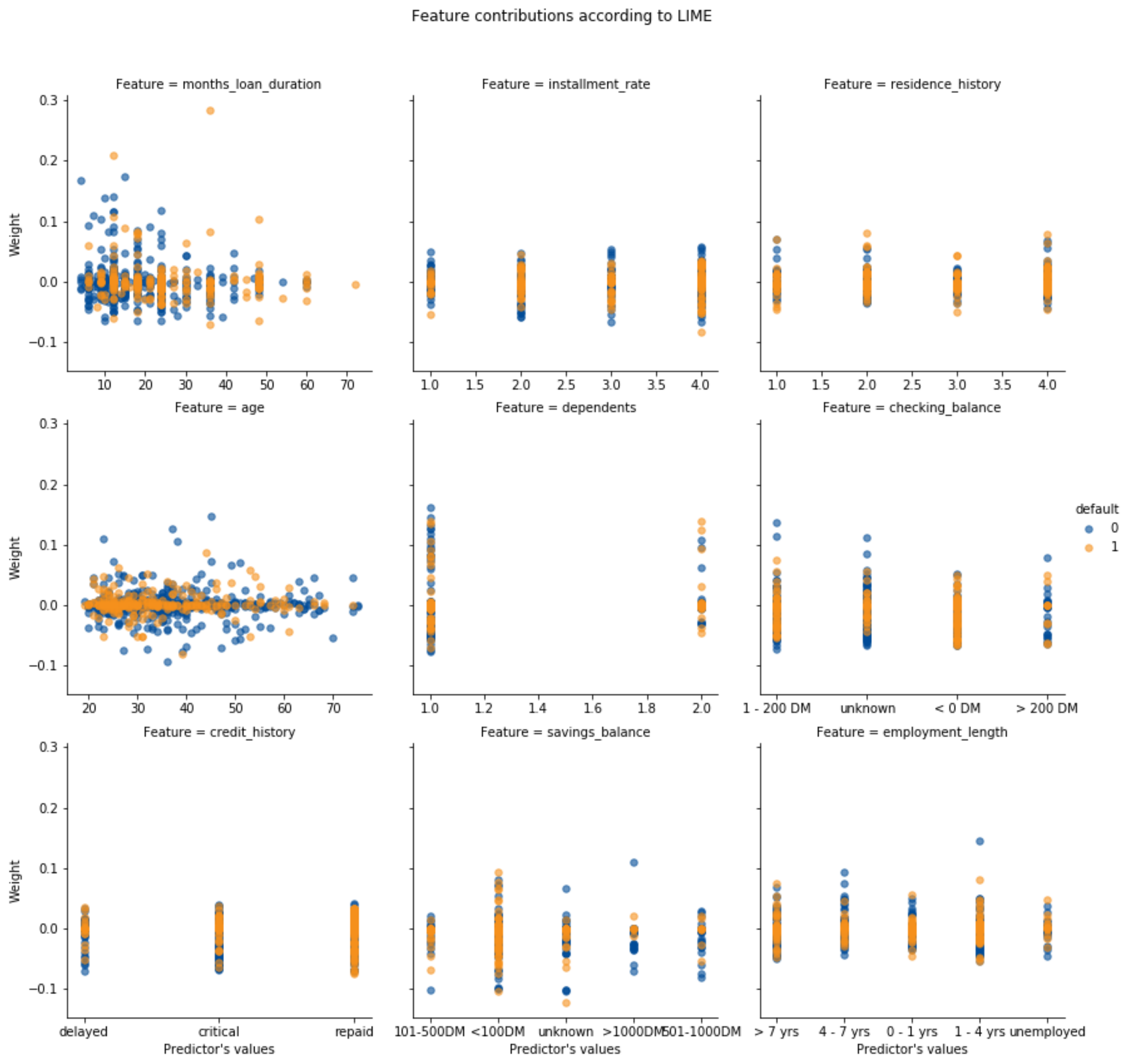
Thanks to the explanation function that fits an interpretable model and generates feature contributions for each instance, it is possible to obtain much more information about the model with this method. By aggregating explanations for all observations from the sample, it is possible to obtain methods in imitation of feature significance and partial dependence, however calculating approach is different. All one needs to do is to collect feature weights for all of the instances calculated with the use of LIME explainer. Then, feature weights are multiplied by the original feature values in order to obtain feature contributions (similarly as in linear statistical models). From the LIME package documentation we can read that the data taken into LIME explainer is scaled, so weights returned by the explainers should be multiplied by the scaled data. When obtaining final contributions calculated with an interpretable model and summing their absolute values withing predictors, overall feature distribution can be plotted as in the figure. Secondly, plotting distribution of each feature contribution and stacking them with original feature values results in sensitivity analysis of predictors and their impact on predictors. When calculating overall feature contribution to the prediction the weights for each predictor have been scaled so that contribution of all explanatory variables is 1. Distribution of feature weights for independent variables have been done for nine most significant regressors obtained in Figure 10. Feature importance computed with LIME weights bring us with the following conclusion: most of the variables visible in the figure are the same as those calculated with Permutation Feature Importance approach. LIME does not include amount, while favors dependents variable Comparing to Permutation Feature Importance the ordering is different. One cannot directly compare weights assigned in this approach with permutation scores as their calculation is absolutely different. LIME assigns higher rank for months loan duration, credit history and employment because it does not recognize any discriminant patterns in amount variable. That conclusion is supported in feature contributions plotted in Figure 11 that illustrates dispersion of the data points weights centered around zero weight for amount variable. Dispersion of true default or non-default clients is marked with orange and blue appropriately. Based on the distribution of default and non-defaults in the figure it is months loan duration as well as checking balance and age discriminates instances pretty well. For the other variables it is not so obvious, as the orange and blue points are stacked with each other. Savings balance weights seem to mirror realistic behavior of the model, assigning observations with higher probability when the balance is unknown or "<100 DM". The rest features weights do not show any logic behind them, especially that their influence on prediction trend is often not consistent with relations in PDP and ALE plots. It points to an important property of the local surrogate models – local fidelity. Local explanations of the model do not have to correctly replicate the model globally. In other words, if features are significant locally (for a small set of observations), this is not true that the features must be important in the global scale. Local approximations matter in case of the local surrogate models.

**Figure 10. Feature Importance results calculated by aggregating absolute values of weights obtained with LIME explainer applied with the training set observations. Values are scaled so that the sum of all feature importances is 1.**



*Source*: own preparation.

**Figure 11. LIME feature contributions for local surrogate model applied over the training set. Each dot represents one instance. Observations being actual default as marked with orange, and non-default with blue . On the vertical axis weight for each instance is shown. Categorical predictors such as savings balance are not ranked with their order, however it does not inflate their contribution interpretation.**



*Source*: own preparation.

Apart from LIME another toolbox has been developed by the authors of this package - submodular pick. It selects a number of instances and presents their explanations. These instances, according to Ribeiro et al. (2016), capture the trustworthiness of the model best. This addresses the issue, that even though the model predicts the instances well (model accuracy is high), the purpose why the model does so might not be in accordance with human logic. What is more, a model user might not have an idea how to capture instances that represent the reliability of an algorithm. Submodular pick does it for us in the following manner: once the explanations for a given dataset are obtained (with $d'$ number of features defined in LIME explainer), features are ranked with their significance ($I_j$ denotes importance of the $j-th$

variable in the explanation space). A predictor $j$ is more significant than predictor $i$ ($I_j > I_i$) if it explains more observations than $i$. Having regressors ordered according to their contribution, submodular pick is assumed to pick up explanations with those important features (starting from the most important ), however it tries to omit instances with exactly the same or similar predictors. For example, suppose a set of $d'$=5 features. First explanation chosen by the toolbox contains feature $x_1$ and $x_3$. By picking another instance submodular pick tries to look for a case with no features $x_1$ and $x_3$, thus maximizing the coverage function.

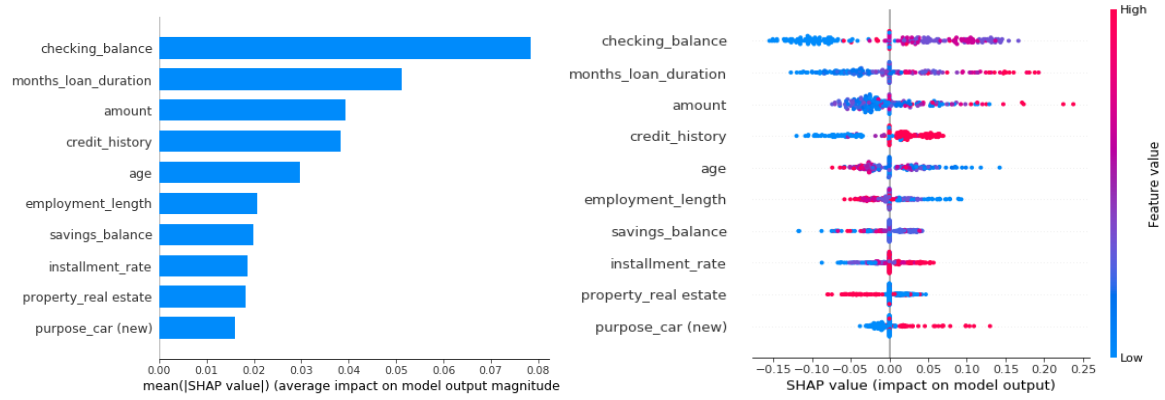
It is often desirable in credit scoring to zoom in on a specified customer and tell what feature attributes regarding personal characteristics or application data are flagged by the model as those contributing positively or negatively to the score. A novel technique involving fitting an interpretable white-box in close vicinity to a single observation has been unarguably a breakthrough in local explanation techniques developed further by many. LIME was presumable first framework developed on the grounds of local surrogate approach. As the goal of LIME is to fit an explainable structure for model predictions, scores produced by the white-boxes are rarely exactly the same as for the black-box one. This bias is strictly dependent on local fidelity of the explanations. Local fitting of a white-box structure to prediction can be manipulated with different parameters, such as proximity measures in LIME explainer, however no standard recommendations have been provided. In addition to that, every interpretable model (especially models from linear class) have some limitations. They are usually not applied for capturing nonlinear structure of the data. They are sensitive to outliers and strong correlation of predictors. Lastly, common linear models must meet some assumptions, which broken, might result in unreliable model outcomes. This stands for another argument against approximating predictions of black-boxes with a white-box model. Although key idea standing behind LIME is disruptive, the way it addresses trust in single model predictions is a matter of discussion. I believe setting LIME parameters properly might bring optimal results, however how do we know whether these explanations can be trusted. How do we know whether observations around the point of interest have been sampled so that they reflect complex dependencies in the data, what proximity measure should be used and what amount of features mirrors true behavior of the classifier. There are many aspects where LIME is susceptible to generating wrong explanations. Although it provides simple and intuitive tool that could make us human better off in explaining individual clients scores, there exist some alternative approaches for local model interpreting.

### *4.3.Shapley values*

Shapley values concept derived from the game theory has been a unified approach enabling peeking into the model in global and local scale. There is a great Python package including a variety of functions for machine learning interpretability calculated with the use of Shapley approach – SHAP (SHapley Additive exPlanations). This package supports both agnostic kernel explainers that can be a model of any time, but also model specific explainers. It supports different Python packages - scikit-learn, TensorFlow, Keras, PyTorch. It deals with different kind of predictions (regression, classification and multiclassification problems). Although computation time is greater as compared to LIME package, for some tree based models some speed up implementations are possible. SHAP enables explaining not only standard tabular data, but also text data and images. This useful toolbox enables inspecting the model from different perspectives. First, agnostic approach. With SHAP plotting significant features in the whole sample is possible. It not only shows average impact of predictors on the outcome, but plots distribution of the variables versus the outcome. Secondly, visualizing impact of individual predictors on predictions can be done with dependence plots. Inspecting the model locally is yet another functionality of the library. SHAP calculates contribution of variables into single predictions. In other words, it explains the model locally. Finally, stacking single observations for the whole dataset (for example training set) is possible. It is done with force plot that shows explanation for all these instances. Authors believe, that combining explanations of single instances allows for global insight into the model and checking whether it can be trusted.

Now a brief visualization of the methods available in SHAP will be presented. As this work aims to present model agnostic solutions, KernelExplainer class from the package has been chosen for the purpose of the model inspection. Except for model agnostic, SHAP offers also model specific explainer dedicated for tree based model or models combining ideas from Integrated Gradients etc. More information can be found in latest SHAP documentation

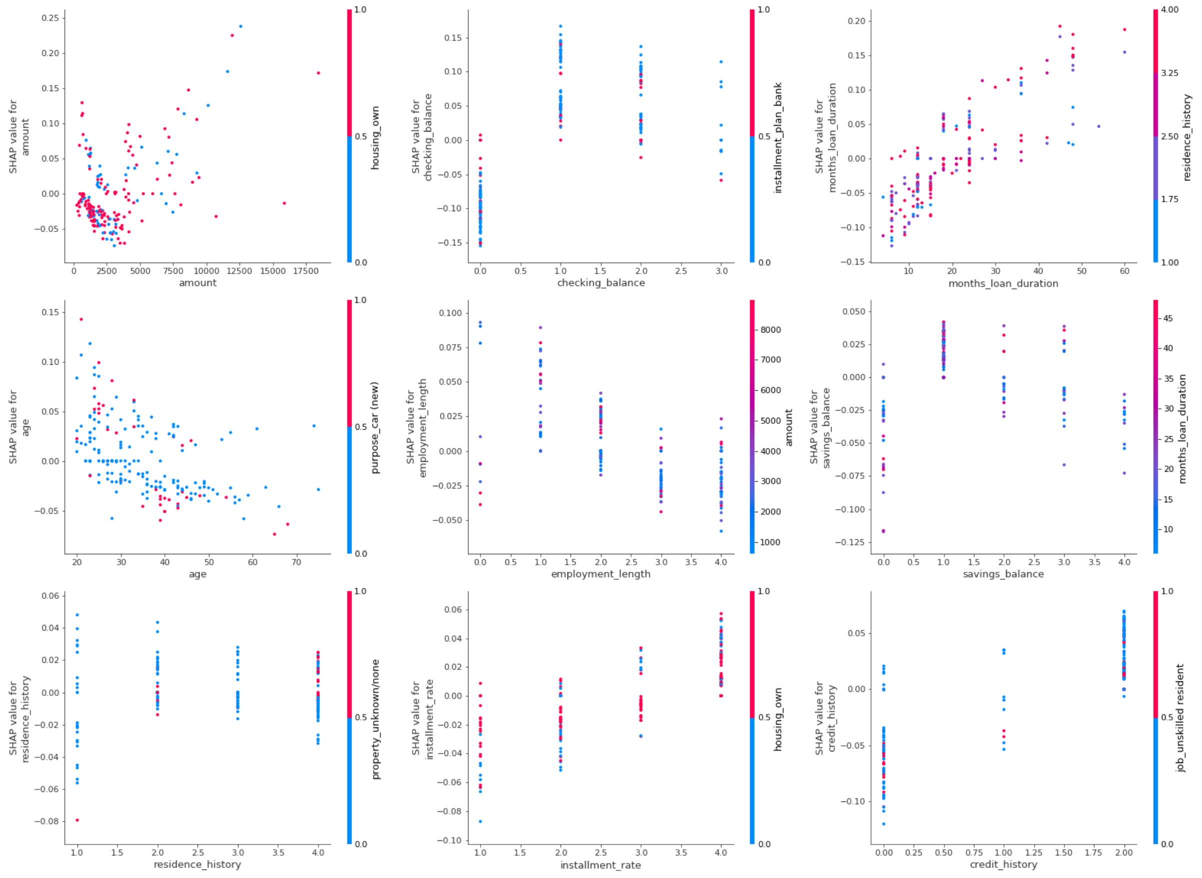
**Figure 12. SHAP feature importance (left panel) and SHAP summary plot (right panel) visualizing feature values contribution. Summary plot represented by beeswarm plot apart from distribution of data points (represented by single dots) informs what is the relationship between impact on model output triggered by low, median or high value of predictor.**



*Source*: own preparation using SHAP Python package.

With summary plots, global insight into the model is obtained. It shows how the given predictor contributes to the prediction on average - in other words, average SHAP value for a given feature. This is for bar plot. A beeswarm plot shows feature contribution for individual instances that in total sum up to the average impact shown in bars. Beeswarm plot is different from the bar plot as it distinguishes the features with respect to their values – positive or negative, so it’s slightly more informative than just simple bar plot. In addition distribution of Shapley values is visible through the overlapping points. For credit history for example beeswarm plot shows that observations with higher value of that feature (credit_history = “repaid”) are assigned with higher SHAP value (contributing positively to the probability of default). For more information about sensitivity of predictors with respect to the regressors SHAP dependence plots should be analyzed.

**Figure 13. SHAP feature contributions called as dependence plot. Their visualization is similar to Partial Dependence or Accumulated Local Effects, however brings more information about model performance – enables capturing meaningful interactions in the data. On the x-axis feature of interest is presented, left-hand vertical axis accounts for calculated Shapley values, and right-hand vertical axis captures explanatory variable that has strongest interaction with predictor chosen.**

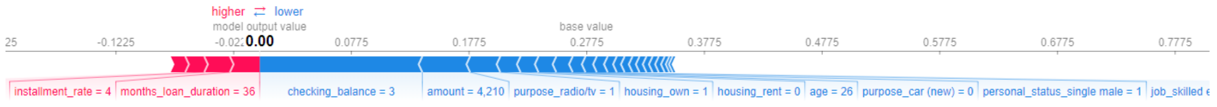


*Source*: own preparation using SHAP Python package.

Dependence plots are very similar to the Partial Dependence Plots and Accumulated Local Effects in their destination. This toolbox has one extra feature – it identifies interaction between distinct features. Apart from revealing the patterns between predictions and predictors' values, dependence plot visualizes the dependency between a chosen predictor and another feature whose interaction with the feature of interest is highest according to calculated Shapley values. This enables verifying whether tree based model indeed capture non-linearity that is so often found in tabular data. (Lundberg and Lee., 2020). Take for example dependence plot for predictor age. In line with theory, the higher SHAP value, the higher contribution of the feature towards the default. According to calculated values there is strong interaction effect with the fact whether client takes the loan for a new car. Suppose he does and purpose_car(new)=1 (red dots in the plot). Younger credit recipients (about age 20 to 30) taking the loan so that they could purchase a new car are more risky as their SHAP values are greater than 0. On the other hand, the same group of clients but of age above 40 years are granted lower scores. Intuition behind those explanations is very logical. It should be remarked that

Model agnostic tools for local explanations are desirable as they enable peeking into any kind of model. However these model-agnostic techniques are often slow and their results differ among calculations. As Lundberg and others (2020) assure, TreeExplainer implemented in SHAP is designed for speed calculations with high precisions. What is more, all meaningful properties of SHAP explanations are preserved – efficiency, linearity, symmetry. Shapley values for single datapoints are designed with "force plots". Length of each feature arrow measures how much a given predictor pushes model prediction from the base value (base is set to the average prediction computed from the set passed to the explainer). Variables with blue force prediction down and those with red force it higher. Bold value points for the regressor actual outcome. As SHAP accepts original training or testing sample with encoded features, categorical variables are shown with the numerical values. Explanation is presented for the same instance as in LIME. Months loan duration equal to 36 and installment rate equal to 4 pushes prediction towards the higher score, however a set of regressors such as checking balance = "> 200 DM", amount = 4210 and purpose_radio/rv=1 and housing_own = 1 drops prediction down. SHAP explanation advocates low score of the model, as the algorithm seems to reason in the following way: credit recipient is a young single male (personal_status_single_male=1) in the productive age (age=26), has own housing, he is a skilled employ so has good job perspectives. The last information is not visible in the figure, however from the explanation object SHAP weights for all predictors have been obtained and job_skilled employee = 1 contributes the lower score. Moreover credit amount is low (4210) and client takes the loan for the purpose of radio/tv purchase. Checking balance is high and is equal to > 200 DM. In accordance with this explanation it seems justified that the model scored the observation with low probability of default.

**Figure 14. SHAP representation of local model inspection Shapley values are computed for an observation from the test set with high prediction error. Model prediction is 0 and average model score is 0.2275. Variables in red support higher probability and blue color features decrease the** score.

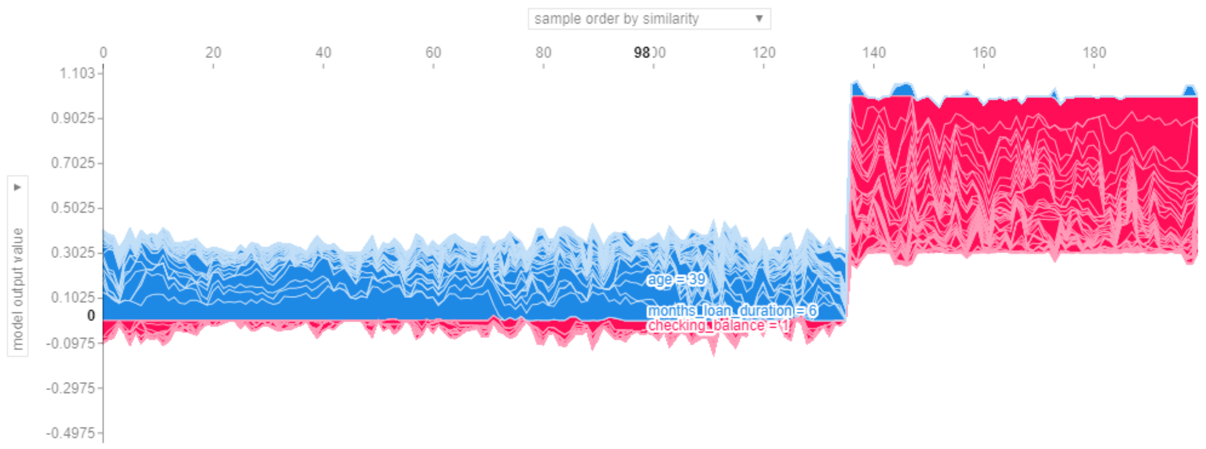


*Source*: own preparation using SHAP Python package.

By obtaining feature contributions for single instances it is possible to measure the strength of features interactions. Plotting these feature impacts for individual samples can help to look at the dependencies from the global scale. And so the force plot (Figure 15) is obtained by stacking single instances explanations together. On the x-scale feature values are plotted. Vertical axis represents the output of the model, however it does not have to be so. Any other feature (dependent or non-dependent) can be obtained on the horizontal or vertical scale by simply changing it with the drop down list. Once again SHAP presents another tool for easy and meaningful interaction detection of the analyzed sample. Looking at single instances feature contributions has yet another advantage. By looking at observations with surprisingly high or low score (detecting them can be done by simply measuring prediction error) previously undetected mistakes in the data can be noticed, as the plot shows exact values of the observations.

Slack et al. (2020) tested empirically that in case of sensitive data application SHAP is a more reliable method than LIME. As both of these frameworks are perturbation based methods, authors underlie that they cannot truly mirror discriminant behavior of the classifiers, however SHAP does it indeed better than LIME in case of sensitive problems (e.g. with highly correlated predictors).

**Figure 15. With obtaining SHAP explanations for single instances and stacking them vertically interactive visualization stated as force plot is constructed. Different combinations of features on vertical and horizontal axis might be presented. By default observations are clustered according their position in a hierarchical clustering. Red feature values support higher probability and blue feature values decrease the score. Plot is interactive, so hovering over it enables showing some features of observations forming the clusters. Visualization of the training set points at two explicit clusters. It is possible to choose any predictor along the vertical and horizontal axis, so for example interaction effect between explanatory variables might be obtained.**



*Source*: own preparation using SHAP Python package.

Explanations based on Shapley values provide not only model agnostic, but powerful methods enabling interpreting the model from both the global and local scale. One does not have to compute Feature Importance or plot Dependence Plots using different approaches, but implement single framework and exploit it efficiently. Many raise the key argument supporting SHAP packages among the others: framework is grounded on solid theoretical background inspired by Lloyd Shapley and satisfies axioms of efficiency, symmetry, additivity and dummy player. They assure that all predictors must contribute fairly to the explanation of an individual, for example. This unified concept distinguishes important regressors in the model, relation between the model outcome and individual predictors' values, captures correlated features, explains single instances asserting local interpretability of the classifier. By implementing SHAP package one can take a closer look at the model predicting a certain event and draw conclusions about its complexity.

Interpretation of LIME and SHAP is different, however both of the tools have similar purpose. They serve for local predictions interpreting. While LIME approximate solutions raise doubts about their results, SHAP provides method based on solid mathematical foundations. One does not have to contemplate on correctness of explanations established with SHAP. In addition to that SHAP is intuitive and according to Lundberg and Lee (2017), consistent with human intuition. This highly informative tool brings the user with methods supporting better understanding of the modelled process.

## 5. Conclusions

Answering the question what features pushed the model to make a certain prediction is crucial in terms of interpretability. It is especially important in systems demanding knowledge about the object or humans whose behavior is being modelled, such as credit scoring, but most importantly about the system itself. If there is some bias in the model because scoring system tends to increase score because of some unreasonable behavior of the clients, machine learning explainability tools should reveal these actions. In addition to that, banks under European supervision are obliged to inform the client why his/her loan application has been rejected. This "right to be informed" (Goodman and Flaxman, 2016) poses another challenge for people considering black-boxes as non-interpretable structures. Nevertheless XAI techniques for local model explanations serve as a solution in this decision process.

Establishing interpretability of the model can be obtained from both the global perspective, but also in local terms, depending on the analyst needs. In particular cases interpretability might be obtained simply by application of single methods. However every entity might establish its own best practices in order to increase trust in decision systems. In order to gain insight into significant model predictors in the whole model, Feature Importance measuring contribution of predictors into the model output should be computed. Important thing is that the data must be labelled, so that prediction error can be calculated. For these purposes both Permutation based Feature Importance or any other technique calculating contribution of input variables might be used. They show how on average the model score is pushed, but they do not reveal direction of that move. This is the very basic method giving insight how the model ranks the features. If more detailed relation between distinct feature values and algorithm response is required, Partial Dependence Plots should be applied to the algorithm predictions. In case problem of collinearity in the data has not been resolved before black-box has been implemented, Accumulated Local Effects should be preferred. Once important features and their relationship with model response function has been known, it would be also reasonable to understand how the model assigns single instances with the scores. In other words, even though predictors significant in the global scale are consistent and reasonable with expert knowledge, assuring model discriminates observations correctly as a whole and also for each observation separately is crucial for enhancing fairness of the model. If the model made a wrong decision when discriminating observations, local explanations based on local surrogate or Shapley value concept help to assure whether decision system took into account attributes compliant with humans. Sometimes methods inspecting small regions visualize shortcomings of the data in predicting false positive instances. It might contribute to pointing that replacing missing data of one attribute would result in a totally different score for example. Visualizing explanations for instances with high prediction error might help to detect outliers in the data that bias results of the model. This kind of local model inspection is known as local variable importance. In addition to that Individual Conditional Expectation curves are also perfect for capturing alternative scenarios. Consider a skilled and employed client assigned with high probability of default. He has not been given a loan now because his salary is low. Assuring with ICE his default score decreases once his salary increases poses a perfect occasion for the future that this man becomes our client.

When picking up solutions for bringing trust in the model one should choose those techniques grounded on solid mathematical foundations, such as Shapley values or at least approaches that have been thoroughly empirically tested. In case of local surrogate implemented in LIME numerical concerns have been raised in the literature about sparse and not always accurate explanations produced with these explainers. Also Permutation based Feature Importance is criticized for inappropriate results in case of highly correlated predictors. However scientific area of machine learning interpretability have been recently broadly investigated and many useful alternatives (often built as an extension to existent techniques) have been proposed. Each of the method presented in this work entail some risk that the explanation of model predictions will be biased, however in most cases they contribute to higher transparency and accountability of decision systems built on the machine learning foundations.

## References:

Aaron Fisher, Cynthia Rudin and Francesca Dominici, 2019. “All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously.” In: arXiv preprint arXiv: 1801.01489.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila and Francisco Herrera, 2019. “Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI.” In: arXiv preprint arXiv: 1910.10045.

Alex Goldstein, Adam Kapelner, Justin Bleich and Emil Pitkin, 2014. “Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation.” In: arXiv preprint arXiv: 1309.6392.

Alvin E. Roth. “The Shapley Value. Essays in Honor of Lloyd S. Shapley, 1988.” Cambridge University Press

Baptiste Gregorutti, Bertrand Michel and Philippe Saint-Pierre, 2017. “Correlation and variable importance in random forests.” Statistics and Computing 27 (3), 659-678.

Brandon M. Greenwell, 2017. “An R Package for Constructing Partial Dependence Plots.” The R Journal 9/1, 421-436.

Bryce Goodman and Seth Flaxman, 2016. “European Union regulations on algorithmic decision-making and a "right to explanation”. In: arXiv preprint arXiv: 1606.08813.

Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib and Thomas Augustin, 2008. Conditional variable importance for random forests. BMC Bioinformatics 9 (1), 307.

Christoph Molnar, 2018. “Interpretable machine learning.”

Daniel W. Apley and Jingyu Zhu, 2019. “Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models.” Northwestern University, USA.

Danilo Bzdok, Naomi Altman and Martin Krzywinski, 2018. “Statistics versus Machine Learning.” Nature Method 15, 233-234.

Debbaut Peter, Andra C. Ghent and Marianna Kudlyak, 2013. “Are Young Borrowers Bad Borrowers? Evidence from the Credit CARD Act of 2009”. Federal Reserve Bank of Richmond Working, 13-09R.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh and Himabindu Lakkaraju, 2020. “Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods.” In: arXiv preprint arXiv: 1911.02508.

Erik Štrumbelj and Igor Kononenko, 2010. "An Efficient Explanation of Individual Classifications using Game Theory." Journal of Machine Learning Research 11, 1-18.

Erik Štrumbelj and Igor Kononenko, 2014. "Explaining prediction models and individual predictions with feature contributions." Knowledge and Information Systems 41, 647–665.

Giles Hooker and Lucas Mentch, 2016. "Quantifying uncertainty in random forests via confidence intervals and hypothesis tests." The Journal of Machine Learning Research, 17 (1), 841 -881.

Giles Hooker and Lucas Mentch, 2019. "Please Stop Permuting Features: An Explanation and Alternative." In: arXiv preprint arXiv: 1905.03151.

Gilles Louppe, Louis Wehenkel, Antonio Sutera and Pierre Geurts, 2013. "Understanding variable importances in forests of randomized trees." NIPS.

Heiko Hotz, 2006. "A short introduction to game theory."

Irina Mihai, Radu Popa and Elena Banu, 2018. The probability of default for private individuals using microeconomic data. What is the role played by macroprudential measures.

Jerome H. Friedman, 2001. "Greedy function approximation: a gradient boosing machine." The Annals of Statistics, 29(5), 1189-1231.

Kjersti Aas, Martin Jullum and Anders Løland, 2020. "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values." In: arXiv preprint arXiv: 1903.10464.

Laura Toloşi and Thomas Lengauer, 2011. "Classification with correlated features: unreliability of feature ranking and solutions." Bioinformatics, 27(14), 1986–1994.

Lehmann, E. L. and J. P. Romano, 2006.. "Testing statistical hypotheses." Springer Science & Business Media

Leo Breiman, 2001. „Random Forests." Machine learning, 45, 5-32.

Leo Breiman, Jerome Friedman, Charles J. Stone and R.A. Olshen, 1984. "Classification and Regression Trees."

Marco Tulio Ribeiro, Sameer Singh and Carlos Guestrin, 2016. "Why Should I Trust You?: Explaining the Predictions of Any Classifier." In: arXiv preprint arXiv: 1602.04938.

Mateusz Staniak and Przemyslaw Biecek, 2018. „Explanations of model predictions with live and breakDown packages." In: arXiv preprint arXiv: 1804.01955.

Michal Polena and Tobias Regner, 2018. Determinants of Borrowers' Default in P2P Lending under Consideration of the Loan Risk Class. Games 9(4), 82.

Milo R. Honegger, 2018. “Shedding light on black box machine learning algorithms.” In: arXiv preprint arXiv: 1808.05054.

Nantian Huang, Guobo Lu and Dianguo Xu, 2016. A Permutation Importance-Based Feature Selection Method for Short-Term Electricity Load Forecasting Using Random Forest. Energies.

Patrick Hall and Navdeep Gill, 2018. An Introduction to Machine Learning Interpretability.” O'Reilly Media.

Prateek Joshi, 2017. “Artificial Intelligence with Python.” Packt Publishing.

Qingyuan Zhao and Trevor Hastie, 2019. “Causal Interpretations of Black-Box Models.” Journal of Business and Economic Statistics.

Scott Lundberg and Su-In Lee, 2017. “A Unified Approach to Interpreting Model Predictions.” In: arXiv preprint arXiv: 1705.07874.

Scott Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal and Su-In Lee, 2020. “From local explanations to global understanding with explainable AI for trees.” Nature Machine Intelligence, 2, 56–67.

Shai Shalev-Shwartz and Shai Ben-David, 2014. “Understanding Machine Learning: From Theory to Algorithms.” Cambrige University Press.

Vijay Arya, Rachel K. E. Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind Samuel C. Hoffman, Stephanie Houde, Q. Vera Liao, Ronny Luss, Aleksandra Mojsilović Sami Mourad, Pablo Pedemonte, Ramya Raghavendra, John Richards, Prasanna Sattigeri Karthikeyan Shanmugam, Moninder Singh, Kush R. Varshney, Dennis Wei, Yunfeng Zhang, 2019. “One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques.” In: arXiv preprint arXiv: 1909.03012.

Wei-Hung Weng, 2019. “Machine Learning for Clinical Predictive Analytics.” In: arXiv preprint arXiv: 1909.09246.

Wojciech Samek, Thomas Wiegand and Klaus-Robert Müller, 2017. “Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models”. In: arXiv preprint arXiv: 1708.08296.

University of Warsaw
Faculty of Economic Sciences
44/50 Długa St.
00-241 Warsaw
www.wne.uw.edu.pl